UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Novedad y diversidad en la recomendación de contactos en redes sociales

Doble Máster en Ingeniería Informática e Investigación e Innovación en Tecnologías de la Información y las Comunicaciones

Autor: Pepa, Sofía Marina
Tutor: Castells Azpilicueta, Pablo

FECHA: Febrero, 2016

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor Pablo Castells por brindarme la oportunidad de poder continuar trabajando con él para la realización del Trabajo de Fin de Máster después de haber finalizado el Trabajo de Fin de Grado hace poco más de un año. Muchas gracias también a todos los compañeros del laboratorio por haber hecho las "horas de trabajo" más amenas (con extra de cotilleo a veces).

Gracias a mi familia, como no, por haberme apoyado siempre, animándome y aconsejándome en los momentos que más lo necesitaba. Jamás me cansaré de repetir que me siento inmensamente afortunada de pertenecer a la familia a la que pertenezco.

También quiero agradecer a mi pareja, Javi, que en este último año y medio me ha soportado y acompañado en el "doloroso" transcurso del doble máster. Gracias por escucharme, comprenderme, motivarme y ayudarme en todo momento, sin ti esto no lo podría haber conseguido.

Por último, me gustaría agradecer a todos aquellos que me han acompañado en estos últimos años, sobre todo a esos amigos que ya eran especiales al terminar la carrera y ahora se han convertido en (espero) amigos para toda la vida. Gracias por esos desayunitos de cotilleos, gracias por las comidas cultas y políticas (en las que nunca participaba), y gracias por todos los bonitos recuerdos que me llevo de esta época.

Ahora sí que pongo punto y final a esta etapa de mi vida. Adiós *colacaos* de la EPS, adiós prácticas, adiós exámenes y adiós a la UAM. Bienvenidas horas de sueño decente, bienvenidos fines de semanas libres, bienvenido tiempo de ocio y bienvenido mundo laboral.

Sofía Marina Pepa

Febrero 2016

Resumen

En el presente trabajo de fin de máster abordamos la recomendación en redes sociales centrándonos en el caso particular de la recomendación de contactos en Twitter, una red social de microblogging que presenta acceso abierto y diversidad de estructuras, tipos de elementos y conexiones.

Centramos nuestro estudio en el planteamiento de nuevas dimensiones de utilidad de la recomendación en red social, explorando nuevos enfoques para realizar la recomendación de contactos donde el objetivo no se centre únicamente en el acierto, sino también en la novedad y la diversidad de la recomendación. Además, planteamos la perspectiva de bien social como nuevo punto de vista para la recomendación, de forma que las recomendaciones no sólo consideren la perspectiva de la utilidad individual para cada usuario, sino que se tenga en cuenta al realizar la recomendación cómo puede influir en ciertas características globales en la evolución de la red.

En el trabajo exploramos además la definición de nuevos algoritmos de recomendación de contactos, basados en la adaptación de modelos probabilísticos de búsqueda y recuperación de información, así como en la adaptación de algoritmos de recomendación que no se conoce que hayan sido aplicados a este nuevo contexto. Asimismo, nos acercamos al planteamiento de algoritmos de reordenación de los contactos recomendados para optimizar ciertas características del grafo, orientadas a la novedad y la diversidad de los contactos recomendados a cada usuario, así como la diversidad estructural de la red.

El trabajo realizado se apoya en la realización de pruebas offline, donde se comparan los algoritmos propuestos junto con otros tomados de la literatura, evaluando en términos de acierto, novedad, diversidad individual y diversidad estructural. Estudiamos además el efecto de la diversidad estructural que promueven los algoritmos en la diversidad de información, mediante la simulación de la propagación de tweets a través de diferentes versiones de la red correspondientes a la acción de diferentes algoritmos de recomendación.

Los resultados de estos experimentos se complementan con pruebas online, donde se somete una selección de los algoritmos anteriores al juicio de usuarios “en vivo”. Para ello, hemos desarrollado una aplicación Android que realiza la tarea de recomendar contactos a los usuarios a partir de la obtención de su subred de Twitter de distancia 2.

Palabras clave

Recomendación, red social, Twitter, novedad, diversidad, bien social, detección de comunidades, modularidad, predicción de enlaces, enlaces débiles

Abstract

The present work delves into the development and evaluation of recommender systems in the context of social networks. In particular, we focus on contact recommendation in Twitter, a microblogging social network which allows open access to data including diverse network structures, item types, and connections.

We research new dimensions of utility in social network recommendation, exploring new ways to recommend users to follow not only focusing on accuracy, but also considering novelty and diversity dimensions of the recommendation. We also explore new social welfare persepectives, where not only the individual user recommendation utility is considered, but also the influence that the recommendation may have on relevant network characteristics.

We explore the definition of new user recommender algorithms as well, based on adaptations of probabilistic information retrieval models, as well as recommender system algorithms that, to the best of our knowledge, were not applied before to this task. We also study the optimization of arbitrary novelty or diversity properties of recommendation by means of recommendation reranking algorithms targetting such properties, focusing on the novelty and diversity of the recommendations delivered to each individual user, and/or the resulting global structural diversity of the network.

We examine our proposed approaches and test our hypotheses by means of offline experiments. We compare in these experiments the effectiveness and properties of our proposed algorithms to each other, and to alternatives from the literature. We test the algorithms for accuracy, novelty, individual diversity and structural diversity. We also study the effect of the structural diversity promoted by the algorithms by simulating the propagation of tweets through different versions of the socila network, corresponding to the action of different recommendation algorithms.

The empirical offline results are complemented with an online test, where a selection of the previous algorithms are put on trial for live users. For this purpose, we have developped an mobile Android application that recommends users baded on the Twitter follow network of distance 2 around each test user.

Keywords

Recommendation, social network, Twitter, novelty, diversity, social welfare, community detection, modularity, link prediction, weak ties

Índice

Índice de tablas

Índice de figuras

1. Introducción

1.1 Motivación

La recomendación está cada vez más presente en tecnología de uso diario relacionada con el acceso a la información y, dicho de un modo abstracto, la selección de opciones entre conjuntos masivos de alternativas. Cada vez se crean más escenarios en los que el usuario tiene acceso a cantidades ingentes de información, haciendo que encontrar información relevante u opciones de interés se convierta en una tarea inabarcable manualmente para el usuario. Como solución a este problema, nacen los sistemas de recomendación, que tomando en cuenta las preferencias de cada usuario, filtran la información y seleccionan aquellos ítems que creen que al usuario le van a interesar, generando así una recomendación.

Los sistemas de recomendación han estado en desarrollo desde principios de los años 90. Amazon fue una empresa pionera en este campo y posiblemente la más popularmente conocida como paradigma en el uso de tecnologías de recomendación. Con el tiempo la inclusión de funcionalidades de recomendación se ha vuelto una práctica común en los sitios Web de comercio electrónico más variados (eBay, Wallmart, Fnac, por citar algunos). En años recientes la recomendación se ha extendido a múltiples campos de aplicación, tales como la recomendación de noticias (Google News), contenido audiovisual (música en Last.fm y Spotify, vídeos en YouTube, películas, series y programas en Netflix o Filmaffinity, o presentaciones compartidas en SlideShare), publicidad personalizada (Google AdSense) o aplicaciones para móviles (Google Play).

Los modelos, algoritmos y metodologías de evaluación que se han desarrollado en este ámbito están típicamente basados en partir de una representación simplificada de la interacción entre usuarios e ítems para generar la recomendación. Esta representación consistente en un valor numérico (o en ocasiones binario), que refleja un grado de interés explícitamente expresado por el usuario respecto a un conjunto de ítems con los que ha interaccionado (por ejemplo, un usuario puntúa de 1 a 5 estrellas cada película que ha visto).

Por otra parte la década de los 2000 se ha caracterizado por un auge masivo de las redes sociales online, y las funcionalidades de recomendación confluyen con ellas con sentido pleno. Recomendar usuarios, utilizar estructura y trazas propias de las redes sociales como dato de entrada para mejorar las recomendaciones, son algunos ejemplos de esta confluencia. De las diferentes perspectivas, nuevas o diferenciadas de las anteriores, que abre la introducción de una red social en el contexto de la recomendación, el presente trabajo se centra específicamente en la recomendación de contactos. La peculiaridad específica que introduce esta tarea respecto a los problemas y soluciones planteados en el campo de la recomendación es el hecho de que, por una parte, los ítems a recomendar se seleccionan entre los mismos usuarios a los que las recomendaciones van dirigidas, y por otra, existe información extra disponible para los algoritmos de recomendación, concretamente los vínculos e interacciones directas entre los usuarios.

Como se documentará en la sección de estado del arte de esta memoria, existen diversos antecedentes en la recomendación de contactos en redes sociales, desde diferentes campos y comunidades investigadoras. Estos antecedentes son no obstante recientes,

o bien previos a las perspectivas actuales de las redes online, por lo que el problema de recomendar contactos está aún ampliamente abierto a la exploración. Por otra parte, observamos la oportunidad de materializar la confluencia de las áreas que inciden en el problema (recomendación, análisis de redes sociales, recuperación de información) más allá de lo que se ha realizado hasta la fecha. Todo ello motiva abordar esta dirección en el presente trabajo.

El trabajo que aquí se expone persigue varias metas: por un lado, realizar una labor de estudio y análisis del estado del arte existente sobre este campo; por otro, explorar la definición e implementación de nuevos algoritmos de recomendación de contactos, y comparar su efectividad y propiedades con algoritmos previamente documentados en la literatura sobre este ámbito; y por último, aportar nuevas perspectivas en cuanto a las dimensiones de evaluación en el contexto de las redes sociales, en particular relacionadas con la novedad y la diversidad de la recomendación (como dimensiones complementarias al acierto), y el beneficio colectivo (como visión complementaria al promedio del beneficio individual). Motiva así este trabajo contribuir a los avances hacia las cuestiones abiertas señaladas.

1.2 Objetivos

Partiendo de una perspectiva del punto en que se encuentra a nivel general y específico, el desarrollo de soluciones y algoritmos para este área, el objetivo general de este trabajo consiste en avanzar hacia nuevas contribuciones en direcciones que el estado del arte no ha resuelto o abordado aún. En particular, el trabajo apunta hacia la toma en cuenta de nuevas perspectivas en el objetivo de la recomendación, tales como la novedad y la diversidad de los contactos recomendados, y no sólo el acierto como objetivo único de la recomendación. También se aborda el efecto de la recomendación desde una perspectiva más global, es decir, cómo influyen diferentes estrategias de recomendación en la evolución de una red (su topología, propiedades, respuesta a estímulos, etc.), más allá del efecto individual en el entorno inmediato de cada usuario, y teniendo en cuenta otras características globales de la red además de su densidad (como el número total de contactos).

El objetivo de este trabajo se particulariza en estudiar la tarea de recomendación de usuarios centrando principalmente el interés en la red social de Twitter. Para ello, el trabajo se estructura en torno a un conjunto de objetivos parciales que se enumeran a continuación:

- Análisis del trabajo relacionado relevante para el tema de estudio, tanto en el contexto específico de Twitter como de otras similares como Facebook, como a un nivel más abstracto sin suponer una red en particular, a fin de elaborar una perspectiva actualizada y exhaustiva que integre bajo un mismo prisma trabajos tanto previos como posteriores a la emergencia de las redes sociales en línea.
- Reproducir y comparar algoritmos documentados en la literatura propios del contexto de la recomendación de usuarios, y adaptar otros que no consta que se hayan aplicado en este contexto. Por ejemplo, algunos algoritmos típicos de recomendación (como basados en vecinos próximos y factorización de matrices) o métodos fundamentales del campo de la Recuperación de Información (como el modelo vectorial, BM25 o los modelos de lenguaje estadísticos).

- Aportar perspectivas novedosas a las cualidades objetivo de la recomendación de contactos. Por un lado, ampliar el enfoque para ir más allá del acierto estudiando métricas de novedad y diversidad de la recomendación, y por otro, introducir la idea de "bien social", de forma que en lugar de optimizar recomendaciones para cada usuario únicamente de forma individual, se consideren óptimos de propiedades globales con el objetivo de conseguir mejorar características de la red en conjunto.
- Analizar y contrastar los resultados de pruebas offline y online, utilizando una descarga de datos de Twitter en el primer caso, y realizando para el segundo caso una aplicación móvil de recomendación de usuarios para Twitter.

Así pues nivel general podemos decir que este TFM apunta a contribuir, de forma inmediata o indirecta, en tres direcciones de avance: por un lado, conseguir mejores algoritmos de recomendación de contactos (empezando por recopilar y comparar soluciones existentes y proponiendo otras nuevas); por otro, plantear y entender mejor qué es lo que aporta valor a una recomendación de este tipo, identificando cualidades relevantes más allá del acierto, y desarrollando medios y técnicas para evaluarlas y optimizarlas; y finalmente apoyar la exploración teórica con pruebas empíricas, complementando un conjunto de experimentos online con el desarrollo y despliegue de una aplicación para realizar pruebas en vivo.

1.3 Estructura del documento

El presente trabajo se estructura en seis capítulos y dos anexos, que se detallan a continuación:

- **Capítulo 1: Introducción.** Definición del contexto, motivación y objetivos del presente trabajo. Se detalla la estructura del mismo y se especifica la notación que se va a utilizar a lo largo de todo el trabajo.
- **Capítulo 2: Estado del arte.** Estudio de la literatura relevante al problema abordado, que abarca tres áreas confluyentes: los sistemas de recomendación en general, la recomendación en redes sociales en particular, y el análisis de redes sociales. Resumiremos de forma general las diferentes familias de algoritmos de recomendación y el progreso de los métodos de evaluación a lo largo de los últimos 20 años; trataremos los antecedentes de la recomendación en el contexto específico de las redes sociales analizando sus particularidades; y recogeremos las nociones y métricas del campo del análisis de las redes sociales que hemos identificado o seleccionado como relevantes para nuestra investigación.
- **Capítulo 3: Algoritmos de recomendación.** Expondremos y describiremos con más detalle los algoritmos de recomendación, tanto del estado del arte como originales, incluidos en el presente trabajo, explicando en qué consisten e indicando la formulación utilizada para el estudio, así como la adaptación, en su caso, a la tarea de recomendar contactos.
- **Capítulo 4: Nuevas dimensiones de utilidad de la red social.** Se detallan las métricas utilizadas para evaluar la calidad de las recomendaciones, tanto desde el punto de vista más tradicional que es el acierto, pasando por otros más novedosos como la novedad y la diversidad, hasta otros abordados por primera vez (hasta donde alcanza nuestro conocimiento) en el presente trabajo.

- **Capítulo 5: Resultados experimentales.** Resultados experimentales, tanto offline como online, así como la interpretación y análisis de los mismos. Se incluye también el detalle de alguno de los barridos realizados sobre aquellos algoritmos con parámetros configurables para obtener su óptimo y en algunos casos justificar la creación de una variación nueva del algoritmo original.
- **Capítulo 6: Conclusiones.** Resumen y contribuciones del trabajo realizado, así como las perspectivas futuras propuestas para la continuación del mismo.
- **Anexo I: Resultados completos offline.** Tablas de resultados de los algoritmos estudiados para el conjunto de datos objetivo con todas las métricas de evaluación, no sólo las principales que se muestran en el apartado 5.
- **Anexo II: Significatividad estadística.** Resultados en forma de tabla de la significatividad estadística los casos en los que se ha realizado para comprobar la validez de los resultados.

1.4 Notación

A lo largo del presente trabajo se van a utilizar la siguiente notación y símbolos.

U	Conjunto de usuarios existentes en el conjunto de datos con el que se está trabajando.
E	Conjunto de enlaces existentes en el conjunto de datos con el que se está trabajando.
E_{train}	Conjunto de enlaces existentes en el conjunto de entrenamiento con el que se está trabajando.
E_{test}	Conjunto de enlaces existentes en el conjunto de test con el que se está trabajando.
Z	Conjunto de todos los aspectos considerados en las medidas de diversidad.
H	Conjunto de hashtags existentes en el conjunto de datos con el que se está trabajando.
C	Conjunto de comunidades identificadas en el conjunto de datos con el que se está trabajando.
$R(u)$	Ránking de contactos recomendados para el usuario *u* en particular.
$g(u)$	Grado del usuario *u*.
$g_{in}(u)$	Grado de entrada del usuario *u*.
$g_{out}(u)$	Grado de salida del usuario *u*.
$N(u)$	Conjunto de usuarios más cercanos (vecindario) que están conectados con el usuario *u* mediante enlaces de entrada y/o salida de *u*.
$N_{in}(u)$	Conjunto de usuarios más cercanos (vecindario) que están conectados con el usuario *u* únicamente mediante enlaces de entrada de *u*.
$N_{out}(u)$	Conjunto de usuarios más cercanos (vecindario) que están conectados con el usuario *u* únicamente mediante enlaces de salida de *u*.

(u, v)	Enlace desde el usuario *u* al usuario *v*.
$weight(u, v)$	Peso del enlace desde el usuario *u* al usuario *v*.
v_k	Contacto recomendado que se encuentra en la posición *k* del ránking.
$f(u, v)$	Función de ránking, calcula la puntuación que el algoritmo de recomendación estima del contacto *v* para el usuario *u*.
$sim(u, v)$	Similitud entre dos usuarios *u* y *v* calculada mediante la función de similitud correspondiente.
$dist(u, v)$	Distancia entre dos usuarios *u* y *v*.
$comm(u)$	Comunidad a la que pertenece el usuario *u*.
$subtopics(u)$	Conjunto de aspectos del usuario *u*.
$u[f]$	Valor del centroide de usuario para una característica *f*.

2. Estado del Arte

El trabajo que se presenta en este documento toca transversalmente varias áreas importantes de la literatura. Se trata de un estudio de la recomendación de contactos en las redes sociales, por lo que el primer ámbito relacionado es naturalmente el de los sistemas de recomendación. Asimismo, por el contexto de aplicación de las recomendaciones que se plantea en nuestra investigación, es relevante para nuestro trabajo el área del análisis de redes sociales, además de los antecedentes específicos en la predicción de links en un grafo. En este trabajo se amplía además la visión estándar de medir la calidad de una recomendación mediante el acierto, prestando atención a otros puntos de vista como la novedad y la diversidad de resultados que aporta al usuario, nociones que se han abordado y desarrollado tanto en el campo de la Recuperación de Información como en el de los sistemas de recomendación. Por último, un punto principal del trabajo consiste en ir más allá y aportar un punto de vista más amplio al medir la utilidad de las recomendaciones, que es la búsqueda de bien social en la red, de forma que en lugar (o además) de recomendar de forma aislada a cada usuario se tenga en cuenta la mejora de ciertas características de la red que se pueden obtener a nivel global.

Cada uno de estos puntos relevantes al trabajo ha tenido un desarrollo previo en antecedentes de diversas áreas que se explican, con nivel selectivo de detalle según su importancia para el presente trabajo, en los siguientes apartados.

2.1 La tarea de recomendación

Los sistemas de recomendación han estado en desarrollo desde principios de los años 90 (Goldberg 1992, Resnick 1994), y se han orientado prioritariamente a la recomendación de productos en comercio electrónico (libros, películas, ropa o cualquier otro artículo en venta en tiendas online como Amazon, eBay o Fnac). En años recientes, se han extendido las tecnologías de recomendación a múltiples campos de aplicación, tales como las noticias (Google News), contenido audiovisual (música en Last.fm y Spotify, vídeos en YouTube, películas y programas en Netflix y Filmaffinity, o presentaciones en SlideShare), publicidad personalizada (Google AdSense) o aplicaciones para móviles (Google Play). Por otra parte, el surgimiento del fenómeno global de las redes sociales, en auge constante desde principios de este siglo, ha dado lugar al planteamiento de escenarios y funcionalidades de recomendación en este nuevo contexto.

Típicamente, la tarea de recomendación consiste en lo siguiente. Se dispone de las preferencias (*ratings*) de los usuarios a los ítems (normalmente representadas como una matriz de interacción usuarios-ítems), las cuales se pueden obtener de forma implícita (por ejemplo, el número de veces que un usuario ha escuchado una canción) o de forma explícita (por ejemplo, puntuaciones que dan los usuarios a películas que han visto). El algoritmo de recomendación selecciona los ítems candidatos (con los que el usuario objetivo no está aún relacionado) y los ordena en una lista (ránking) de forma descendiente según la relevancia que el algoritmo ha estimado que cada uno tendrá para el usuario. Esta tarea se realiza de forma individual para cada usuario, y principalmente se busca tener la mayor cantidad de aciertos en la recomendación realizada a cada uno.

En los siguientes sub-apartados se explican dos de los aspectos esenciales presentes en toda tarea de recomendación: los algoritmos que generan las recomendaciones a cada usuario y los métodos de evaluación para medir y comparar su efectividad.

2.1.1 Algoritmos de recomendación

En la literatura hay infinitud de algoritmos de recomendación que se basan en diferentes principios y técnicas (Adomavicius 2005, Ricci 2001) para la generación de recomendaciones. Todos parten de los mismos puntos y tienen el mismo objetivo, diferenciándose únicamente en la forma de elegir y ordenar los ítems a recomendar.

La familia de algoritmos más importante es la de filtrado colaborativo, que se caracteriza porque en la recomendación a un usuario se utiliza conocimiento sobre otros usuarios, de tal forma que los usuarios aprovechan la experiencia unos de otros (Adomavicius 2005). Las ventajas del filtrado colaborativo son principalmente dos: buenos niveles de acierto (en términos de error de predicción), así como de novedad y de diversidad; y se pueden recomendar ítems sin necesidad de disponer de ninguna descripción de los mismos, sino sólo ratings de usuarios. Las estrategias de filtrado colaborativo adolecen de la limitación llamada de arranque en frío, es decir, tienen problemas para generar recomendaciones para usuarios poco activos o ítems poco conocidos.

Dentro de los algoritmos de filtrado colaborativo destacan los basados en vecinos próximos y los de factorización de matrices. Los algoritmos basados en vecinos próximos infieren preferencias de un usuario basadas en las de usuarios (vecinos) similares, o bien infieren preferencias por ítems similares a los que se ha observado que el usuario prefiere, donde en todos los casos la similitud se mide en términos de la interacción entre usuarios e ítems. Los algoritmos de factorización de matrices, por otro lado, consisten en descomponer la matriz original de ratings en un producto de varias matrices (típicamente 2 o 3) de forma que se generan k factores latentes que establecen un espacio de características común, tanto para los usuarios como para los ítems, permitiendo la comparación directa entre ellos.

Otra familia importante, distinta del filtrado colaborativo, es la de los algoritmos basados en contenido, que generan recomendaciones mediante la comparación del contenido que describe cada ítem y el contenido que le interesa al usuario objetivo (Adomavicius 2005). Los métodos basados en contenido pueden recomendar ítems nuevos o poco conocidos, aunque tienen en general la misma dificultad que el filtrado colaborativo para usuarios con poca actividad, ya que no se conoce lo suficiente sobre sus gustos. La principal limitación de la recomendación basada en contenido es que, recomendando ítems parecidos a aquéllos por los que el usuario mostró preferencia, se puede producir un cierto efecto de "encasillamiento" para éste, con poca novedad y diversidad en la recomendación.

Las soluciones de recomendación más efectivas en la práctica suelen ser los algoritmos híbridos, que combinan métodos de filtrado colaborativo y basados en contenido (Adomavicius 2005). Existen diferentes técnicas para realizar esta combinación, donde destacan el método monolítico (aplicar el recomendador sobre datos de ratings y contenido combinados), el paralelismo de varios sistemas (combinación ponderada de los resultados obtenidos de diferentes recomendadores), y la implementación segmentada (un recomendador pre-procesa el input del siguiente a ejecutar). Las soluciones híbridas permiten, por ejemplo, paliar el problema del arranque en frío propio del filtrado colaborativo, o la recomendación de ítems sin descripción, propio de los algoritmos basados en contenido, gracias a que unos algoritmos respaldan las debilidades de los otros.

2.1.2 Evaluación de sistemas de recomendación

Hay múltiples formas de medir la calidad de recomendación generada por un algoritmo, dependiendo de lo que se busque optimizar. Dentro de esta variedad es útil distinguir, en primer término, entre las métricas de error y las métricas de ránking.

Las métricas de error se centran en las puntuaciones que el recomendador predice, de forma que miden qué tan cerca ha estado la puntuación predicha respecto a la puntuación real que el usuario ha dado al ítem objetivo. Las métricas comunes de este conjunto son *Mean Absolute Error* (MAE), definida como la media de la diferencia absoluta entre los ratings predichos y los reales; y *Root Mean Squared Error* (RMSE), igual que la anterior pero tomando el cuadrado de las diferencias (para acentuar los valores más grandes) y la raíz cuadrada del promedio.

Por otro lado, las métricas de ránking sólo prestan atención al orden en el que los ítems han sido recomendados, independientemente de la puntuación que el algoritmo ha calculado. Para este fin se toman métricas que desde hace tiempo se vienen desarrollando en el campo de la Recuperación de Información para evaluar sistemas de búsqueda. Dentro de este conjunto cabe distinguir tres tipos principales de métricas, dependiendo del objetivo que se busque optimizar:

- **Acierto**: evalúan el grado de acierto de la recomendación realizada por el algoritmo. Observa los primeros ítems recomendados por el algoritmo y calculan la proporción de ítems recomendados que gustan al usuario. Entre estas métricas se encuentran Precisión, *Recall* y nDCG, entre muchas otras.
- **Diversidad**: busca que la recomendación incluya ítems que no sean demasiado parecidos o redundantes entre sí. Algunas métricas de diversidad son ERR-IA, *Subtopic Recall*, o *Intra-List Dissimilarity* (ILD).
- **Novedad**: valora recomiéndela recomendación de ítems que es probable que el usuario no conozca. En esta línea se han propuesto métricas como *Expected Popularity Complement* (EPC) o *Expected Profile Distance* (EPD).

Mientras las métricas de precisión han sido ampliamente estudiadas en la literatura, las de novedad y diversidad son más innovadoras. Por ello, se ha tomado como principal referencia el capítulo realizado por Castells et al. referente a la novedad y la diversidad en sistemas de recomendación (Castells 2015).

En el apartado 4 se detallarán éstas y el resto de dimensiones de evaluación utilizadas, así como su formulación implementada para la evaluación de los algoritmos en el presente trabajo.

2.2 Recomendación social

La recomendación en redes sociales introduce dos aspectos distintivos respecto a escenarios anteriores típicos en los sistemas de recomendación. En primer lugar, además de las preferencias que puedan tener los usuarios hacia ítems de la plataforma (como darle a “me gusta” a páginas de Facebook o hacer *retweets* en Twitter), el sistema dispone de una red en la que los usuarios están explícitamente conectados entre sí mediante vínculos simétricos (como amigos en Facebook) o asimétricos (como *followers* en Twitter). Por otra parte, en una red social cabe recomendar, además de ítems (*posts*, música, etc.), contactos. Es decir, los usuarios pueden ser no sólo destinatarios de las recomendaciones, sino también objeto de las mismas.

2.2.1 Recomendación de ítems

En los campos de aplicación tradicionales, la recomendación de ítems se realiza prestando atención únicamente a las preferencias que tienen los usuarios sobre los ítems que conoce. Así, destacan las recomendaciones que se realizan por ejemplo en Amazon o Netflix, donde no existe una red social asociada a la plataforma. Sin embargo, en aplicaciones en las que se dispone de una red social se pueden aprovechar las estructuras de red para mejorar las recomendaciones de ítems.

En años recientes han proliferado trabajos documentados en la literatura donde se recomiendan ítems a los usuarios de una plataforma que cuenta con una red social propia. Un ejemplo es el trabajo realizado por Zhang et al. (Zhang 2013), donde tratan la recomendación de productos a usuarios en Epinions[1] y Ciao[2], ambas páginas en la que los usuarios puntúan y dan su opinión sobre diferentes tipos de productos. En este caso, se propone un *framework* que detecta comunidades por tópicos para construir dominios interpretables para filtrado colaborativo por dominios, y se realiza la recomendación mediante factorización de matrices teniendo en cuenta la topología de la red social. Otro ejemplo parecido es (Aranda 2007), que realiza la recomendación de juegos en la página BoardGamesGeek[3] utilizando los ratings de los usuarios y las amistades entre ellos.

Asimismo, en (Liu 2010) se plantea que los algoritmos de filtrado colaborativo no distinguen amigos dentro de un vecindario frente a desconocidos con gustos parecidos, por lo que se podrían mejorar estos algoritmos incorporándoles las conexiones existentes entre usuarios según la red social. En este caso, las pruebas se realizan sobre la plataforma Cyworld[4], una página de compartición de fotos y *blogging* (tipo Facebook) muy popular en Corea del Sur.

Además del uso de datos explícitos de preferencia usuario-ítem junto a la red social, existen otros estudios donde se utilizan datos de preferencia implícita. Es el caso de la recomendación de canciones a usuarios en Last.fm[5], plataforma para escuchar música, donde se usan las frecuencias de escucha de canciones por usuarios, junto con las conexiones sociales entre usuarios (Konstas 2009). En este trabajo, los autores comparan un algoritmo de *random walk* frente a otros de filtrado colaborativo, y concluyen que el primero obtiene mejores resultados en este contexto.

En estos trabajos se realiza la tarea de recomendación de ítems a usuarios igual que algoritmos tradicionales, utilizando datos explícitos o implícitos para establecer preferencias de los usuarios hacia los ítems, pero apoyan esta información con datos de la topología de la red existente entre los usuarios. Para el presente estudio, sin embargo, el objetivo no es utilizar la topología de la red como apoyo, sino que se busca que sea uno de los elementos principales en los que se basen las recomendaciones.

Otros artículos se enfocan a resolver otros problemas además de la recomendación de usuarios. En (Ma 2008), el objetivo es resolver los problemas de escalabilidad y mejorar la recomendación a aquellos usuarios con muy pocas interacciones. En (Clements 2010) se considera un nuevo espacio, los *tags*, que complementan al de usuarios e ítems para realizar recomendaciones sobre cualquiera de estos tres espacios. Por último, en (Bonhard 2006) se va un paso más allá y se propone, en lugar de utilizar una única red

[1] Epinions, http://www.epinions.com/
[2] Ciao, http://www.ciao.es/
[3] BoardGamesGeek, https://boardgamegeek.com/
[4] Cyworld, http://www.cyworld.com/
[5] Last.fm, http://www.last.fm/

de conexiones entre usuarios para mejorar las recomendaciones, unir varias redes, como puede ser la de Facebook y la de MovieLens, para conseguir más información de los mismos y poder mejorar aún más las recomendaciones.

Todos estos estudios tratan con mucha información procedente de diferentes fuentes. Se tratan de lecturas de interés teórico, pero se escapa de los límites del presente trabajo estudiar y desarrollar ideas en esta línea.

Mientras en la recomendación de ítems existe una separación entre los elementos a los que se destina la recomendación (usuarios) y los que se recomiendan (los ítems), en la recomendación de usuarios esta separación se unifica, de forma que los destinatarios y los objetos de la recomendación son los mismos: los usuarios.

2.2.2 Recomendación de contactos

Numerosos estudios modernos se han documentado en la literatura en los que se adaptan los algoritmos de recomendación de ítems para recomendar contactos en redes sociales. De particular importancia para el presente trabajo son los precedentes que se han centrado en la red social de Twitter, por lo que un buen punto de partida es el *survey* (Kywe 2012) en el que se hace un resumen de la literatura hasta el 2012 sobre los tipos de recomendación en Twitter (de *followees*, *followers*, *hashtags*, *tweets* y noticias).

Dentro de la literatura de recomendación de contactos en Twitter, destaca en primer lugar (Gupta 2013), que describe la primera versión del servicio de recomendación "Who To Follow" de Twitter. En este artículo, se explica por un lado la arquitectura utilizada para este servicio (*Cassovary*, un motor de procesamiento de grafos en memoria desarrollado por ellos mismos), y algunos de los algoritmos novedosos que han utilizado para recomendar usuarios, como una adaptación que realizan del algoritmo SALSA (Lempel 2001) utilizando la red ego del usuario. Se trata de un trabajo que guarda gran relación con el estudio que vamos a plantear, por lo que es uno de los que más importancia tiene dentro del estado del arte.

Un artículo relacionado a este realizado un par de años antes es (Kim 2011), donde se plantea un sistema de recomendación de usuarios y tweets para Twitter aplicando un modelo probabilístico y utilizando tanto los tweets del usuario objetivo como las relaciones con otros usuarios que éste tuviese. Este artículo se presentó en un momento en el que Twitter no realizaba estas tareas de recomendación, de forma que en su momento aportaron novedad en este campo.

Asimismo, se encuentra el artículo de Hannon de un año antes a este último (Hannon 2010). Se propone la herramienta *Twittomender* creada para la recomendación de usuarios y tweets según la red social de Twitter. En este caso, se hace énfasis en las aproximaciones basadas en contenido (como el algoritmo de Rocchio), estudiando diferentes formas de representar al usuario, esto es, según sus tweets, los tweets de sus followers, de sus followees, o de ambos. Aplicando entonces una versión particular de tf-idf, realizan una poda donde se quedan con los 20 términos con mayor tf-idf para cada usuario. Los resultados obtenidos indican que Rocchio con el centroide extendido a los tweets de los followers mejora a usar sólo los tweets propios (aunque es de destacar que el conjunto de datos utilizado para los experimentos no tiene una separación limpia entre entrenamiento y test, además de que la partición deja muy poca información en test). Se trata de una investigación que aporta muchas ideas al presente trabajo, donde destaca la de cuál es la mejor forma de representar al usuario, pero no respecto a tweets sino a la propia topología de la red.

Por último, es de destacar un *position paper* realizado por Golder et al. (Golder 2009), donde se describen los mecanismos existentes hasta el momento para realizar la recomendación de usuarios, y plantean diferentes enfoques para realizar esta misma tarea, todo desde el contexto de la red social de Twitter. Más concretamente, se detallan diferentes principios de recomendación que se pueden utilizar para la recomendación de usuarios, como la reciprocidad, intereses comunes, audiencias compartidas, o por filtros de personas. Se trata de un trabajo de lectura interesante, pero aporta más una visión de contexto que de ideas a aplicar.

Como extra, además de los artículos que tratan el conjunto de datos de Twitter, se ha buscado en la literatura otros que tratan redes sociales similares, como el realizado por Huang et al. (Huang 2013), donde se presenta el algoritmo de recomendación que sigue LinkedIn (red social asimétrica como Twitter) para realizar las recomendaciones de usuarios mediante el servicio llamado "People you may know". Tras realizar un análisis sobre el efecto de la diversidad de la estructura en la tasa de aceptación de las recomendaciones de los usuarios, se concluye con la idea de que una alta densidad de conexiones entre usuarios junto con una baja densidad de la estructura da lugar a una alta tasa de relevancia en las recomendaciones. Trata una idea planteada en varias investigaciones que se desarrollará a lo largo del presente trabajo, que es analizar los enlaces existentes entre los usuarios así como las posibles funciones que cumplen en la red.

2.2.3 Predicción de links

La recomendación de usuarios encuentra también precedente directo en una tarea que se viene abordando en el campo de las redes sociales y la ciencia de grafos desde hace tiempo: la predicción de links en un grafo. Esta tarea ha sido objeto de estudio por muchos campos desde principios de los 90 y equivale en el ámbito de la recomendación de usuarios a crear nuevas conexiones entre usuarios que predeciblemente interesaría que exista. De todos los artículos de investigación encontrados en la literatura sobre la tarea de predicción de links, destacan principalmente tres como líneas relevantes para el presente estudio.

En primer lugar, Liben-Nowell et al. (Liben-Nowell 2003) definen varios métodos para realizar recomendaciones planteándolo como un problema de predicción de links (más concretamente, basándose en medidas sobre la proximidad entre los usuarios). Comparan algoritmos basados en vecindarios (vecinos comunes, Jaccard, Adamic, etc.) y en el conjunto de caminos entre usuarios (Katz, PageRank, SimRank, etc.), junto otros métodos de nivel más alto (clustering). Como resultado, observan que la topología de la red por sí sola no contiene información útil, mientras que Katz y algunos de los algoritmos más simples (vecinos comunes y Adamic) consiguen resultados consistentemente buenos. Se trata de un referente para este trabajo, ya que de esta investigación realizada por Liben-Nowell et al. se extraen numerosos algoritmos que pueden resultar interesantes para someter a comparación.

Otro artículo interesante es (Kashima 2009), donde se aborda el problema de la predicción de links y formula un nuevo algoritmo semi-supervisado para resolverlo. Se basa en la utilización de dos características del grafo: la fuerza de los links que unen a los usuarios mediante la técnica de propagación de etiquetas, y la similitud entre usuarios mediante los métodos de similitud de Kronecker. Los resultaos obtenidos sobre los conjuntos de datos sobre los que se ha aplicado son interesantes, de forma que este artículo concluye con la idea de que el algoritmo podría ser una gran promesa como método de resolución al problema de predicción de links y el trabajo futuro a realizar.

Por último, es de destacar también (Lichtenwalter 2010), que propone otro método semi-supervisado para la resolución del problema, pero que utiliza aspectos de la red que no se habían explorado hasta el momento, como la distancia del grafo y el desequilibrio del mismo. Los resultados observados mejoran en gran medida el *baseline* tomado, un algoritmo no-supervisado llamado *PropFlow* basado en el conocido sistema del caminante aleatorio.

2.3 Análisis de redes sociales

Junto con los objetivos de acierto, diversidad y novedad en la recomendación, la perspectiva de bien social (*social welfare*) que se aborda en este TFM plantea tener en cuenta las propiedades globales hacia las que puede evolucionar la estructura de la red social y la incidencia que los algoritmos de recomendación pueden tener, intencionadamente o no, en ello. Este planteamiento requiere definir qué propiedades estructurales de las redes son relevantes y cómo se pueden analizar. A este respecto empezamos por considerar los fundamentos ampliamente estudiados en la literatura del análisis de grafos y redes sociales que proporcionan técnicas y métricas para analizar la estructura de una red, y que tienen interés desde el punto de vista de la utilidad potencial que se deriva de ellas en nuestro contexto, en particular por su conexión con dimensiones relacionadas con la novedad y diversidad. Entre estos fundamentos enfocamos nuestra atención en las nociones relacionadas con el coeficiente de clustering, los enlaces débiles, y las comunidades, que pasamos a analizar en la subsecciones que siguen.

La idea del uso que investigaremos de las métricas de grafos será, como veremos más adelante, observar y medir cómo varía el valor de las métricas si se añaden a la red las conexiones que propone un algoritmo de recomendación.

2.3.1 Coeficiente de clustering

La diversidad significa ausencia de redundancia, por lo que aquellas métricas que pueden servir como medida de redundancia pueden utilizarse de la misma forma para medir la diversidad de la recomendación. Así, una recomendación será más diversa cuanto más bajo sea el coeficiente de clustering del grafo resultante. El coeficiente de clustering global de un grafo se define como la tasa de caminos de longitud dos que están cerrados:

$$C(G) = \frac{número\ de\ caminos\ cerrados\ de\ longitud\ 2}{número\ de\ caminos\ de\ longitud\ 2}$$

Otra definición alternativa de esta métrica consiste en el promedio de los coeficientes de clustering de los nodos del grafo, donde la métrica para un nodo se define como la tasa de pares de vecinos del nodo que están conectados entre sí. En general las dos definiciones tienden a concordar, aunque pueden divergir en algunos casos. No obstante en nuestros experimentos hemos visto que las dos nociones dan resultados más o menos equivalentes, por lo que en lo que se documenta en esta memoria nos restringiremos a la definición indicada más arriba.

2.3.2 Enlaces fuertes y débiles

Desde los años 70 se ha estudiado el valor potencial de los llamados enlaces débiles, donde destaca la teoría de Granovetter (Granovetter 1973) que argumenta que la coordinación social se ve más influenciada por las relaciones sociales entre usuarios con poca relación (enlaces débiles) que por relaciones que son más cercanas como puede ser la familia o amigos (enlaces fuertes). Por ello, se plantea el posible valor que puedan tener algoritmos que fomenten la presencia de enlaces débiles entre sus recomendaciones.

La noción de enlace débil es abierta y admite diferentes definiciones. Granovetter propone la noción de enlaces llamados "puente", basados en el análisis de componentes conexas y medidas de distancia. Así, se define puente global como un arco cuya eliminación daría lugar a la separación de una componente conexa en dos. Desde el punto de vista de la recomendación, recomendar un enlace débil según esta noción sería recomendar un contacto de una componente conexa distinta a la componente del usuario al que la recomendación va destinada. Las redes sociales online típicas suelen contar en la práctica con una componente gigante fuertemente conexa que abarca bastante más del 90% de la red, por lo que esta definición resulta demasiado restrictiva para nuestros propósitos, pues son pocos los usuarios externos a la componente gigante disponibles para recomendar enlaces débiles, y típicamente se trata de regiones residuales o poco activas de la red, cuya utilidad es menos clara.

Granovetter plantea también la noción de "puente local", que define como aquellos arcos cuya eliminación incrementa en estrictamente más de dos la distancia entre los nodos que conecta. Existe una noción que generaliza este concepto, que es simplemente la noción de arraigo de un arco (meter cita), y que se define como el solapamiento del vecindario de los nodos que el arco conecta:

$$A'(u,v) = Jaccard(N(u) - \{v\}, N(v) - \{u\})$$

Los puentes locales son simplemente arcos con arraigo cero. En nuestros experimentos hemos comprobado que el número de puentes locales de un grafo correlaciona empíricamente con el arraigo promedio de sus arcos. Ello nos hace considerar más útil y general la noción de arraigo que la de puente local, y de hecho captura más información que una noción booleana de distinguir entre arraigo cero o mayor que cero.

Si bien la noción de componente conexa es demasiado estricta en el contexto de las redes sociales debido al fenómeno habitual de la componente gigante, es bien cierto que se pueden típicamente identificar subconjuntos menos triviales de usuarios que forman comunidades aunque no completamente inconexas, sí claramente diferenciadas e identificables, basada por ejemplo en densidades locales de enlaces, u otras propiedades comunes a los nodos del subconjunto. Consideramos pues la opción de contemplar la definición de arco débil como aquél que conecta dos subconjuntos de la red, pero relajando la noción de componente conexa estricta a la de comunidad. De Meo et al analizan recientemente esta misma idea en el contexto de la red social de Facebook (De Meo 2014). En su investigación definen enlace débil como aquel que une dos usuarios de diferentes comunidades, y como enlace fuerte, el que conecta a dos usuarios de la misma comunidad.

2.3.3 Comunidades

Las comunidades son una unidad natural que se observa en las sociedades: las personas se dividen en grupos según sus intereses, ocupación, edad, etc. y por ende en las redes sociales. Es interesante que la definición de la estructura de comunidades pueda ser simplificada a una cuestión observable en la estructura de los enlaces del grafo, sin que por ello sea trivial su utilidad para el análisis.

La detección de comunidades es una de las sub-áreas más amplias del análisis de redes sociales y la ciencia de grafos en general. Existe una gran cantidad de algoritmos centrados en la detección de comunidades, y la inmensa mayoría definen la partición de comunidades de la misma forma, como aquella que maximiza la modularidad.

La modularidad es una métrica que mide qué tan fuerte es la división de los usuarios de un grafo en comunidades, de forma que cuanto más alto es su valor, más clara, pura y fuerte será la unión entre usuarios de la misma comunidad y, de la misma forma, la separación entre las comunidades. La modularidad se define por la siguiente fórmula:

$$Q = \frac{1}{2m}\sum_{ij}\left[w_{ij} - \frac{g(v_i)g(v_j)}{2m}\right]\delta(c_i, c_j)$$

Donde m es la suma de los pesos de los enlaces del grafo, w_{ij} es el número de enlaces existentes entre los nodos v_i y v_j, $g(v_i)$ es el grado del nodo v_i, c_i es la comunidad a la que pertenece el nodo v_i y δ es la función delta de Kronecker (vale 1 si $c_i = c_j$ y 0 en caso contrario).

Encontrar la división que maximiza la modularidad es un problema NP-completo, por lo que los algoritmos de detección de comunidades realizan aproximaciones a la solución óptima para realizar esta tarea. Según el proceso seguido por estos algoritmos, se pueden clasificar en cinco tipos: basados en la centralidad de los enlaces, en la optimización avara de la modularidad, en la representación matricial del grafo, en *random walks*, y basados en la teoría de la información.

Para este trabajo, se han seleccionado como objeto de estudio los algoritmos de detección de comunidades más destacados y populares en la literatura, que se resumen a continuación.

Louvain method

Se trata de un algoritmo heurístico basado en la optimización avara de la modularidad (Blondel 2008) y su funcionamiento se divide en dos fases principales (Figura 1). En la primera fase, se identifican pequeñas comunidades optimizando la modularidad de forma local en los nodos (es decir, agrupando los nodos según la densidad de los enlaces entre ellos). En la segunda fase, se construye una nueva red cuyos nodos son las comunidades que se han encontrado en la primera fase. Al finalizar la segunda fase, se vuelve a repetir el proceso utilizando esta vez el grafo modificado, y la ejecución finaliza cuando la modularidad respecto al grafo original no mejora.

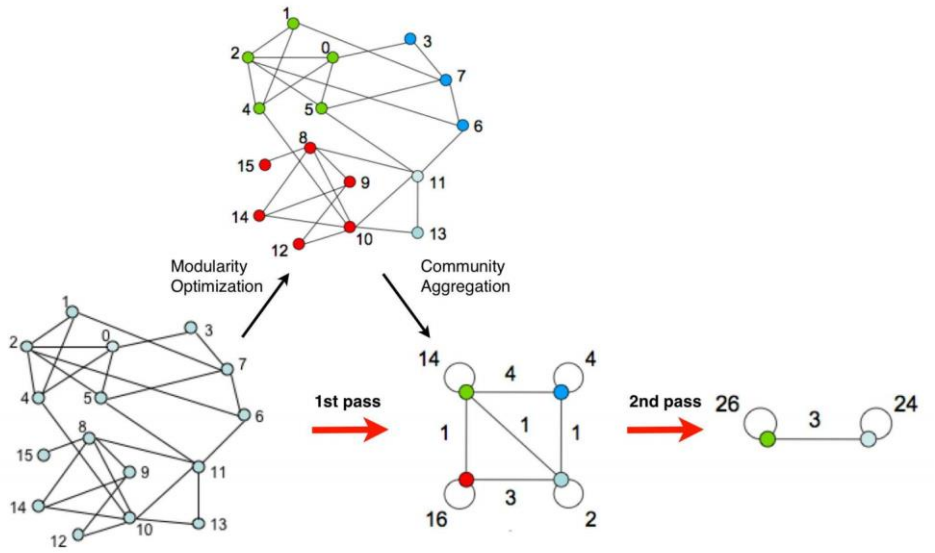

Figura 1. Diagrama del funcionamiento del algoritmo de Louvain para la detección de comunidades en un grafo (Fuente: Blondel 2008).

La ganancia de modularidad obtenida al mover un nodo v_i a una comunidad C se calcula mediante la siguiente fórmula extraída del trabajo original. Como se puede observar, se trata en esencia de la diferencia entre las modularidades, pero se plantea de esta forma por cuestiones de eficiencia.

$$\Delta Q_{louvain} = \left[\frac{\sum_{in} + k_{v_i,in}}{2m} - \left(\frac{\sum_{tot} + k_{v_i}}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_{v_i}}{2m}\right)^2\right]$$

Donde $\sum_{in}$ es la suma de los pesos de los enlaces dentro de la comunidad C, $\sum_{tot}$ es la suma de los pesos de los enlaces incidentes a nodos de la comunidad C, k_{v_i} es la suma de los pesos de los enlaces incidentes al nodo v_i, y $k_{v_i,in}$ es la suma de los pesos de los enlaces de v_i a nodos en la comunidad C.

Infomap

Pertenece al conjunto de algoritmos que toman ventaja de la estructura de la comunidad con el fin de representar a la red utilizando menos información que la codificada en la matriz de adyacencia completa (Rosvall 2008).

La estructura de la comunidad se representa según una nomenclatura de dos niveles basado en la codificación de Huffman (Huffman 1952): uno para distinguir las comunidades de la red, y otro para distinguir los nodos en una comunidad. El problema de encontrar la mejor partición se expresa como minimizar la cantidad de información necesaria para representar algún *random walk* en la red utilizando esta nomenclatura.

Con una partición que contiene pocos enlaces hacia otras comunidades, un caminante aleatorio probablemente se quedaría más tiempo dentro de estas comunidades, por lo que sólo se necesitaría el segundo nivel para describir su trayectoria, consiguiendo una representación compacta, que es el objetivo.

La fórmula objetivo está entonces compuesta por dos términos: el primero da el número medio de bits necesarios para describir un movimiento entre dos comunidades, y el segundo, para uno dentro de la misma comunidad.

$$L(M) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p_{\circlearrowleft}^{i} H(P^{i})$$

Donde M es una partición de un conjunto de n nodos en m comunidades, $q_{\curvearrowright}$ es la suma de las probabilidades en cada paso de que un caminante aleatorio salga de cada uno de los módulos existentes, $H(Q)$ es la entropía de los movimientos entre comunidades, $p_{\circlearrowleft}^{i}$ es la probabilidad en cada paso de que un caminante aleatorio siga en la comunidad i, y $H(P^{i})$ es la entropía de los movimientos dentro de la comunidad i.

Leading Vector Algorithm

Este algoritmo utiliza la técnica espectral para maximizar la modularidad del grafo (Newman 2006), que consiste en encontrar de forma iterativa la división en dos del grafo (y de cada comunidad en las siguientes iteraciones) que maximice la modularidad. Para ello, a partir de la fórmula de la modularidad se extrae la definición de la matriz de modularidad, que es la siguiente:

$$B_{ij} = A_{ij} - \frac{g(v_i)g(v_j)}{2|E|}$$

Donde A_{ij} es el número de enlaces entre los nodos v_i y v_j.

La separación en sub-grupos depende del signo del autovector y autovalor que se obtiene en la matriz de modularidad para cada nodo, de forma que en cada iteración se busca la partición que maximiza la siguiente fórmula:

$$Q_{newman} = \frac{1}{4|E|} \sum_{i=1}^{|V|} (u_i^T s)^2 \lambda_i$$

Donde λ_i y u_i^T son el autovalor y el autovector del nodo v_i, y s es un vector columna que vale 1 o -1 dependiendo si v_i pertenece al grupo 1 o al grupo 2 en que se va a dividir la comunidad.

Girvan-Newman

Se trata de un algoritmo sencillo para detectar comunidades en un grafo de forma jerárquica (Girvan-Newman 2004). Utiliza el arraigo de los enlaces, y su funcionamiento consiste encontrar el enlace con el mayor valor de arraigo, eliminarlo del grafo, y recalcular el arraigo del resto de los enlaces. Este proceso se repite hasta que no quedan más enlaces.

De esta forma, cada vez se elimina el enlace con más probabilidad de unir a dos usuarios de la misma comunidad (los de arraigo más alto), y al final el progreso se obtiene un dendograma que muestra el orden en el que se ha partido el grafo (Figura 2).

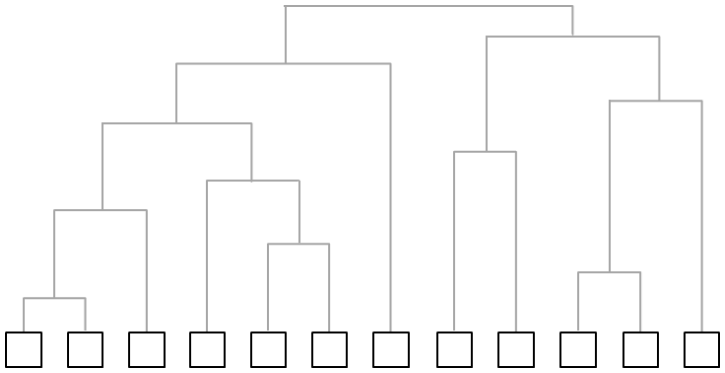

Figura 2. Dendograma resultado del algoritmo de Girvan-Newman.

Una vez se dispone de este gráfico jerárquico, se selecciona la profundidad a la que se quiere cortar la jerarquía para formar las comunidades (Figura 3). El corte se realiza en función al número de comunidades que se desea manejar, de forma que si se corta arriba del todo sólo hay una comunidad, y si se corta abajo del todo hay una comunidad para cada nodo.

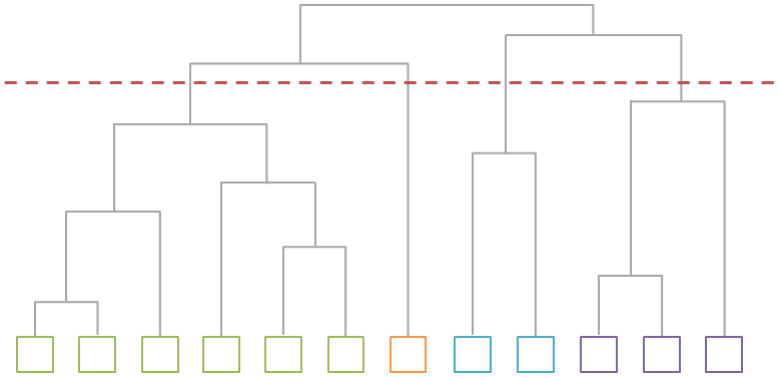

Figura 3. Ejemplo de corte en el dendograma de Girvan-Newman.

Markov Cluster Algorithm

Se basa en la simulación de recorridos estocásticos en el grafo objetivo para encontrar la separación óptima en comunidades (Van Dongen 2000). El algoritmo funciona de la siguiente forma:

1. Se crea la matriz de adyacencia A donde el elemento A_{ij} será 1 si existe un enlace del nodo v_i al v_j, o cero en caso contrario (opcionalmente se pueden añadir autoenlaces para cada nodo).
2. La matriz de adyacencia A se normaliza por columnas (la suma de todos los valores de cada columna tiene que ser igual a 1).
3. Se eleva la matriz de adyacencia A normalizada a la potencia e (parámetro de entrada de este algoritmo).
4. Cada columna de la matriz resultante se reajusta según el parámetro r (otro parámetro de entrada de este algoritmo), de forma que cada elemento de cada columna se eleva a la potencia r y la columna se vuelve a normalizar.
5. Se repite los pasos 3 y 4 hasta encontrar la convergencia en la matriz, es decir, obtener una matriz estacionaria (cuyo estado ya no varía con las iteraciones).

Para interpretar las comunidades resultantes, se dividen los nodos en dos tipos: atractores, que atraen a los nodos que tienen un valor positivo en su fila de la matriz obtenida, y los que son atraídos por los atractores. Para formar las comunidades,

simplemente se agrupan los nodos atractores con los nodos que atraen (no suele ser común que un nodo pertenezca a más de una comunidad).

Su funcionamiento intuitivo se puede ver en forma de imagen (Figura 4) o de animación[6], ambas realizadas por su creador y que pone a disposición junto con información del algoritmo en su página web[7].

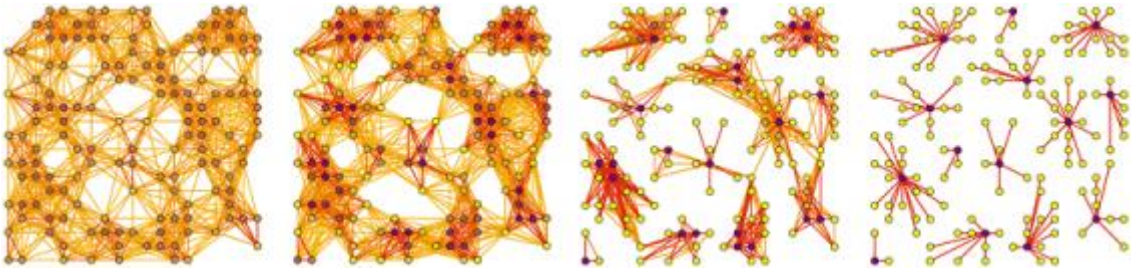

Figura 4. Representación de cuatro iteraciones del algoritmo de Markov Cluster para la detección de comunidades en un grafo (Fuente: Van Dongen 2000).

El algoritmo escala muy bien según el tamaño del grafo, y suele obtener buenos resultados en cuanto a la separación por comunidades, destacando su robustez frente al ruido en los datos.

Para realizar una valoración de estos algoritmos, Orman realiza un trabajo de investigación donde en total compara 11 algoritmos de detección de comunidades (Orman 2011), entre los que se encuentran los descritos en este capítulo. Somete los algoritmos a diferentes pruebas para probar la efectividad así como la escalabilidad que presenta cada uno de los algoritmos.

Como resultado final, se observa que en todos los casos Infomap supera al resto de algoritmos analizados en cuanto al éxito de la identificación de las comunidades. Otros algoritmos que consiguen buenos resultados (aunque no tan buenos como Infomap) son Markov Cluster y Louvain, y por el contrario, Leading Eigenvector se ve muy superado por el resto de algoritmos. En cuanto a la escalabilidad, para los casos en los que se ha probado (redes de hasta 5000 nodos) los algoritmos de Infomap y Louvain funcionan mejor cuanto más grande es la red, Leading Eigenvector funciona peor en estos casos, y Markov Cluster y Girvan-Newman funciona igual sin importar el tamaño de la red.

A pesar de que se traten de pruebas de laboratorio que respecto a los datos tratados en el presente trabajo se quedan un poco pequeños, este estudio facilita una valoración interesante respecto a los algoritmos de detección de comunidades elegidos para este estudio a realizar.

.

[6] Animación del algoritmo MCL (http://www.micans.org/mcl/ani/mcl-animation.html)
[7] Clustering mediante el método de MCL (http://www.micans.org/mcl/)

3. Algoritmos de recomendación

La incorporación de funcionalidades de recomendación de contactos en redes sociales online ha empezado a materializarse hace relativamente pocos años (Gupta 2013, Hannon 2010) por lo que el desarrollo y comparación de algoritmos efectivos es aún un campo abierto. Numerosos algoritmos específicamente orientados a redes sociales están documentados en la literatura, y hemos implementado en este trabajo los más representativos y/o relevantes para, tras realizar un pequeño análisis de la variación de los parámetros y configuración de cada uno, someterlos a una comparativa desde diferentes dimensiones de evaluación, con otros algoritmos que investigamos aquí. En particular, investigamos la adaptación de algoritmos de campos adyacentes, concretamente orientados a la recomendación de ítems con propósito general, y orientados a la búsqueda y recuperación de información. La adaptación de estos algoritmos a la recomendación de contactos no se ha documentado previamente en la literatura y forma parte en ese sentido de la aportación original del presente trabajo.

A continuación se definen con detalle y por grupos los algoritmos que se han incluido en la investigación.

3.1 Algoritmos triviales

A modo de referencia de extremos o aproximaciones simples, incluimos en la comparativa algunos métodos de recomendación más o menos triviales, que pasamos a describir.

Popularidad

Se trata de un algoritmo que recomienda contactos por orden descendente del número de seguidores de los usuarios (es decir, según el grado de entrada de los nodos, ya que de salida no tendría sentido). Es ésta una recomendación no personalizada, pues a todos los usuarios se recomienda el mismo ránking, excepto por la pequeña diferencia de que se evita recomendar a cada usuario los contactos con los que ya tiene relación. La función de ránking de esta simple aproximación es:

$$f(u, v) = g_{in}(v)$$

Donde $g_{in}(v)$ representa el número de seguidores del usuario v (enlaces entrantes al nodo). La API de Twitter, por ejemplo, proporciona este dato, pero en nuestros experimentos utilizaremos el número de seguidores dentro del subgrafo que creemos mediante la descarga de Twitter para los experimentos.

Este método, a pesar de su extrema simplicidad, es uno de los más extendidos en escenarios reales: es el que se nos muestra en las listas de "más vendidos", vídeos con más visitas en Youtube, tweets más retweeteados, programas de TV con más audiencia, ránking de taquilla en cine, *best-sellers*, etc.

Random

Este algoritmo recomienda contactos de forma aleatoria a cada usuario, de nuevo omitiendo aquellos que éste ya tenga.

$$f(u, v) = random()$$

La efectividad de la recomendación aleatoria, medida con una cierta métrica, se puede interpretar como la esperanza priori del valor de la métrica cuando a cada usuario se le recomiendan todos los demás usuarios, y depende generalmente de la densidad de los datos. Es decir, a mayor tasa de pares usuario-ítem con rating observado (en el conjunto de test, para ser más precisos), mayor es el valor de la métrica de la recomendación aleatoria.

FOAF (Friend Of A Friend)

Este método se fija en el vecindario común de los usuarios candidatos y del objetivo, y aplica el principio de recomendar a los usuarios aquellos usuarios con más contactos en común. Intuitivamente, intenta recomendar aquellos "amigos de amigos", es decir, los enlaces transitivos. La coincidencia de contactos se puede medir de diferentes maneras, principalmente en número total (intersección), o el número relativo (ratio entre la intersección y la unión de los contactos de cada usuario). Así pues contemplaremos estas dos funciones de ránking.

a) Intersección:

$$f(u,v) = |N(u) \cap N(v)|$$

b) Jaccard:

$$f(u,v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} = \frac{|N(u) \cap N(v)|}{|N(u)| + |N(v)| - |N(u) \cap N(v)|}$$

La diferencia entre la intersección y Jaccard es que la primera no se normaliza respecto a la unión, de forma que si por ejemplo un usuario candidato tiene como vecindario al resto de usuarios (salvo el objetivo), la intersección con el usuario objetivo será la máxima posible. Mientras que la intersección recomendaría ese contacto el primero, Jaccard lo penalizaría por el tamaño de la unión de los vecindarios y podría incluso no ser recomendado.

3.2 Algoritmos de predicción de links

La literatura específica en recomendación de contactos es amplia, ya sea planteada como tal, o bajo la denominación de "predicción de links". Entre las propuestas del área seleccionamos para nuestro estudio las que siguen, por su efectividad, representatividad, y/o uso en contextos comerciales.

Adamic

El algoritmo propuesto por Adamic et al. (Adamic 2003) establece la similitud entre dos usuarios según el número de ítems que tienen en común, siendo los ítems contenido de los perfiles de usuario, como pueden ser las palabras que utiliza cada usuario para describirse (por ejemplo "informático", "deportista", "cine", etc.). Además, ítems que son poco comunes tienen más peso que otros muy comunes, ya que se utiliza el logaritmo invertido de la frecuencia del ítem.

$$sim(A,B) = \sum_{i \in A \cap B} \frac{1}{\log frec(i)}$$

Donde $frec(i)$ se entiende como el número de veces que aparece el ítem i en las descripciones de todos los usuarios en conjunto.

Sin embargo, Kleinberg y Liben-Nowell proponen una adaptación al contexto objetivo de este estudio, interpretando los ítems como vecinos en común, y la frecuencia de un ítem como el tamaño del vecindario del usuario (Liben-Nowell 2003 2003). De esta forma, con este algoritmo se estudia la importancia de los vecinos en común que tiene el usuario objetivo y el contacto a recomendarle.

$$f(u,v) = \sum_{w \in N(u) \cap N(v)} \frac{1}{\log g(w)}$$

El planteamiento es muy similar al algoritmo de FOAF, pero en este caso se añade el matiz de la importancia de los vecinos comunes que, como se verá más adelante, muestra ser de gran utilidad para obtener en general mejores resultados empíricos.

Closure

En inglés "cierre", recomienda los contactos de segundo grado que además tengan enlace al usuario objetivo. En otras palabras, si u tiene enlace a v, v tiene enlace a w y w tiene enlace a u, entonces al usuario u se le recomienda el contacto w.

$$f(u,v) = \begin{cases} |N(u) \cap N(v)|, & si\ (u,v) \in E_{train} \\ 0, & en\ otro\ caso \end{cases}$$

Este algoritmo sólo tiene sentido en un entorno donde los enlaces entre usuarios son de tipo dirigido, ya que debe existir un enlace de w a u pero en ningún caso de u a w, pues no se puede recomendar un contacto que el usuario ya tiene. Por ello, en aplicaciones como Facebook o LinkedIn, este algoritmo no podría aplicarse, pero sí en redes dirigidas como la de Twitter.

PageRank

PageRank (Page 1998) es uno de los algoritmos más conocidos entre los algoritmos basados en *random walk*. Utiliza la estructura de enlaces de un grafo para establecer la importancia que tiene cada nodo en el mismo. Básicamente, se basa en que el volumen de enlaces entrantes es un indicio de importancia, pero que además la importancia de un enlace entrante aumenta si procede de otro usuario importante (recursividad), a la vez que la importancia de un enlace disminuye en la medida en que el usuario del que procede tenga muchos enlaces de salida.

$$pr_0(u) = \frac{r}{|U|}$$

$$pr(u) = r + (1-r) \sum_{v:(v,u) \in E_{train}} \frac{pr(v)\ weight(v,u)}{g_{out}(v)} + \left(1 - \frac{\sum_{w \in U} pr(w)}{|U|}\right)$$

$$f(u,v) = pr(v)$$

Donde $pr_0(u)$ es el valor inicial de pagerank del usuario *u*, y es necesario definir un parámetro r que representa la probabilidad de teleportación del caminante aleatorio.

Se trata, como resulta además obvio en la definición de la función de ránking, de un algoritmo cuyas recomendaciones no dependen del usuario al que vayan dirigidas. Por ello, junto con esta versión estándar de PageRank (no personalizada), consideramos asimismo una variación personalizada basada en la versión propuesta por White & Smyth (White 2003). Mientras que en la versión básica todos los usuarios tienen una probabilidad de teleportación de r independiente del usuario objetivo de la recomendación, en la versión personalizada esta probabilidad varía entre el usuario objetivo y el

resto, en particular, la teleportación sólo se puede realizar hacia el usuario objetivo. Así, además de que los usuarios adquieran importancia según la estructura de enlaces, también lo harán según la cercanía al usuario objetivo.

$$ppr_0(u) = 1$$

$$ppr(u) = \text{ini}(u) + (1-r) \sum_{v:(v,u)\in E} \frac{ppr(v)\, weight(v,u)}{g_{out}(v)} + \left(1 - \frac{\sum_{w\in U} ppr(w)}{|U|}\right)$$

$$\text{ini}(u) = \begin{cases} r\,|U|, & si\ es\ el\ usuario\ objetivo \\ 0, & en\ otro\ caso \end{cases}$$

$$f(u,v) = ppr(v)$$

Finalmente, proponemos otra versión más equivalente a la anterior pero con un par de variaciones añadidas, que respondían al objetivo de realizar una personalización más "extrema". En primer lugar, se omiten los enlaces que van desde los usuarios hacia el objetivo, y segundo lugar, al llegar a un sumidero se obliga a volver directamente al usuario objetivo. De esta forma, se consigue eliminar el posible sesgo introducido por la diferencia de vecindarios de los usuarios objetivo, de forma que se les recomiende en igualdad de condiciones.

$$pppr_0(u) = \begin{cases} r, & si\ es\ el\ usuario\ objetivo \\ \dfrac{1-r}{|U|-1}, & en\ otro\ caso \end{cases}$$

$$pppr(u) = \begin{cases} r + (1-r) \displaystyle\sum_{s:g_{out}(s)=0} pppr(s), & si\ es\ el\ usuario\ objetivo \\ (1-r) \displaystyle\sum_{v:(v,u)\in E_{train}} \dfrac{pppr(v)}{|N_{out}(v)|}, & en\ otro\ caso \end{cases}$$

$$f(u,v) = pppr(v)$$

SALSA (Stochastic Approach for Link-Structure Analysis)

Pertenece también a la familia de algoritmos de *randomwalk*, y en este caso se ha utilizado la adaptación hecha por trabajadores de Twitter implementada para la recomendación de usuarios con la herramienta *Who To Follow* (Gupta 2013). De forma intuitiva, lo que este algoritmo plantea es que un usuario u tiende a tener un enlace hacia aquellos usuarios que tienen enlaces provenientes de usuarios similares a u. En grafos no dirigidos, este algoritmo simplemente busca cerrar triángulos entre usuarios.

Primero se construye un grafo bipartito personalizado para cada usuario donde se diferencian dos partes, que denominan en el artículo por distinguirlas como "izquierda" y "derecha". En la parte izquierda, se encuentren usuarios del círculo de confianza del usuario objetivo (obtenidos mediante *randomwalk* egocéntrico sobre él), llamados "*hubs*"; y en la parte derecha, usuarios hacia los que los de la parte izquierda tienen enlace, llamados "autoridades".

Se trata de un algoritmo iterativo que, a diferencia del resto de algoritmos de *randomwalk*, realiza dos pasos en cada iteración, uno hacia adelante y otro hacia atrás (o viceversa). Así, se asignan *scores* a ambas partes, obteniendo usuarios que recomendar en la parte derecha, y usuarios similares al usuario objetivo en la izquierda (que también pueden recomendarse según el principio de homofilia).

A continuación se muestran las fórmulas correspondientes al cálculo de las preferencias de los contactos para ambos lados del grafo bipartito. La fórmula correspondiente al lado de los hubs se denota con $h(u)$ y el de las autoridades, con $a(u)$.

$$h(u) = \frac{|\mathbf{h}|}{|H|} * \frac{g_{out}(u)}{|\mathbf{e}|}$$

$$a(u) = \frac{|\mathbf{a}|}{|A|} * \frac{g_{in}(u)}{|\mathbf{e}|}$$

Donde $|\mathbf{h}|$ ($|\mathbf{a}|$) es el número de nodos del grafo bipartito que son hubs (autoridades); $|H|$ ($|A|$), la cantidad de usuarios del grafo completo que pueden ser considerados hubs (autoridades), es decir, los que tienen al menos un enlace de salida (entrada); y por último $|\mathbf{e}|$, el número de enlaces que hay presentes en el grafo bipartito entre los hubs y las autoridades.

3.3 Algoritmos clásicos de recomendación

Más allá de los algoritmos específicamente diseñados en la literatura para la recomendación de contactos, cabe plantearse la aplicación de algoritmos típicos de recomendación de ítems a la recomendación de contactos, simplemente "plegando" el espacio de ítems sobre el espacio de usuarios. Es decir, considerar la matriz del grafo como una matriz usuario-ítem, sólo que el conjunto de ítems es a la vez el conjunto de usuarios.

En la literatura de los sistemas de recomendación se han propuesto innumerables métodos y algoritmos, y como veremos en los experimentos, su adaptación directa a una red social produce buenos resultados empíricos, a pesar de que esta dirección ha sido muy poco explorada aún en la literatura. De entre los múltiples algoritmos del área, hemos seleccionados los que por un motivo u otro podemos considerar más representativos y/o resultan más efectivos. Los describimos a continuación.

Vecinos próximos (kNN)

Los algoritmos de vecinos próximos (kNN) son una de las estrategias de filtrado colaborativo (uso de información de unos usuarios para producir recomendaciones a otros) más extendidas en el campo de los sistemas de recomendación. Seleccionan los *k* vecinos más similares al usuario (kNN orientado a usuario) o al ítem (kNN orientado a ítem) objetivo, de forma que mediante la combinación lineal del rating de los vecinos se realiza una predicción de rating para el usuario objetivo. Se obtiene entonces el ránking final de ítems ordenando de forma descendente estas predicciones (Adomavicius 2005). Para el ámbito de la recomendación de usuarios en el que se centra este estudio, basta con entender los ítems como contactos, que son los mismos usuarios, de forma que el rating de un usuario a otro vale 1 si existe enlace del primero hacia el segundo.

Existe una gran cantidad de variaciones para estos algoritmos, ya que tienen muchos parámetros configurables: el número *k* de vecinos, la similitud mínima a considerar como relevante, o el número de ítems en común para una similitud válida.

Los algoritmos de vecinos próximos se han desarrollado en dos perspectivas posibles: recomendación de vecinos próximos por usuario (User kNN) y por ítem (Item kNN). En el primer caso, se recomiendan los contactos que "han gustado" a usuarios similares al objetivo; y en el segundo, se recomiendan los contactos que se parecen a los contactos que le "han gustado" al usuario objetivo. A continuación se muestran las fór-

mulas para kNN correspondientes a las versiones orientadas a usuario e ítem respectivamente.

$$f(u,v) = \sum_{\substack{w \in N(u) \\ (w,v) \in E}} cos(u,w)\, weight(w,v)$$

$$f(u,v) = \sum_{\substack{w \in N(v) \\ (u,w) \in E}} cos(v,w)\, weight(u,w)$$

Para calcular la similitud entre dos usuarios, se puede utilizar varios métodos (Jaccard, Pearson, etc.), entre los cuales escogeremos la más común: la similitud mediante coseno (sin normalizar). Este método mide el ángulo entre los vectores de usuarios y establece que son más similares aquellos cuyos vectores tengan la orientación más parecida. Su fórmula es la siguiente:

$$cos(\vec{a}, \vec{b}) = \frac{\sum_c a_c b_c}{\sqrt{\sum_c a_c^2}\ \sqrt{\sum_c b_c^2}}$$

Donde a_c es el valor de la preferencia del ítem *c* al usuario *a*. Por ejemplo, a_c podría ser la puntuación que el usuario *a* le ha puesto a la película *c*.

De esta forma, tomando como vector de usuario el peso de los enlaces que tiene hacia otros usuarios, la fórmula final de similitud de usuarios mediante coseno queda de la siguiente forma.

$$cos(u,v) = \frac{\sum_{w:\,(u,w),(v,w) \in E_{train}} weight(u,w)\, weight(v,w)}{\sqrt{\sum_{w:(u,w) \in E_{train}} weight(u,w)^2}\ \sqrt{\sum_{w(v,w) \in E_{train}} weight(v,w)^2}}$$

Factorización de matrices

La factorización de matrices es un método de filtrado colaborativo basado en modelo que de forma general busca descubrir los factores latentes que explican los datos observados, es decir, qué hace que a un usuario le guste un ítem (Koren 2009, Hofmann 1999, Sarwar 2000). Para aflorar este espacio latente, se genera una representación propia (modelo) descomponiendo mediante procesos matemáticos la matriz de ratings original en el producto de otras más pequeñas. El uso de factores latentes reduce notablemente la dimensionalidad del problema, y además no es necesario concretar su significado para aplicarlos al cálculo de preferencias de usuarios e ítems.

Para nuestros experimentos hemos seleccionado el algoritmo de recomendación de Hu et al. (Hu 2008) como representante de la familia de algoritmos de factorización de matrices, dada su efectividad empírica y su popularidad en el área. Su particularidad reside en que para calcular los vectores no se utiliza directamente el rating r_{ui}, sino un valor que representa la "confianza" de observarlo. Este valor se denota con c_{ui} y como se observa en su fórmula a continuación, depende de un parámetro α.

$$c_{ui} = 1 + \alpha r_{ui}$$

Este algoritmo divide la matriz de ratings en otras tres mediante el proceso de optimización por mínimos cuadrados, que busca minimizar la siguiente función para obtener los vectores de usuario (x_u) y los de ítem (y_i):

$$\min_{x_*, y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_{u \in U} \|x_u\|^2 + \sum_{i \in I} \|y_i\|^2 \right)$$

Donde p_{ui} es el rating binarizado e I es el conjunto de ítems de la colección. Nótese que el segundo sumando de la fórmula se añade para que no se produzca un fenómeno de *overfitting* de los datos.

Una vez se dispone de los vectores que relacionan cada usuario e ítem con cada factor latente, la función que calcula la preferencia ($\hat{p}_{ui}$) se obtiene prácticamente de forma inmediata realizando el producto entre los vectores obtenidos de la minimización anterior.

$$\hat{p}_{ui} = y_i^T x_u$$

Al igual que en los algoritmos de vecinos próximos, en este contexto se entienden los ítems como los mismos usuarios y la fórmula de ránking final replanteada es la siguiente:

$$f(u, v) = y_v^T x_u$$

Rocchio

Se trata de un algoritmo basado en contenido, que utiliza la descripción de los contactos para realizar la recomendación (es decir, sin tener en cuenta la información de otros usuarios). Dada una representación de los ítems mediante un vector de características, Rocchio consiste en el cálculo de centroides de los conjuntos de ítems por los que cada usuario ha mostrado preferencia, de forma que se obtiene para cada usuario un vector propio que representa su relación con cada característica (*feature*) de los mismos (Adomavicius 2005). Una fórmula común para el cálculo de los centroides es la siguiente:

$$u[f] = \frac{1}{|\mathbf{u}|} \sum_{i:r(u,i)\neq\emptyset} r(u,i)\ i[f], \text{ donde } \mathbf{u} = \{r(u,i) \neq \emptyset | i \in I\}$$

Donde $u[f]$ denota el valor del centroide de usuario para una característica *f*, $i[f]$ el valor del vector de ítem para la característica *f*, y $r(u,i)$ es la preferencia del usuario *u* al ítem *i*. El valor de $i[f]$ se toma normalmente como $tfidf(f,i)$ para medir la importancia que tiene la característica en el ítem y en la colección.

En el contexto específico de la red social de Twitter con la que se va a tratar en el presente trabajo, se han tomado como ítems los tweets creados por cada uno de los usuarios, siendo entonces las palabras presentes en estos tweets las características. En cuanto a la interpretación de los ratings de usuarios a tweets, se han establecido diferentes preferencias según la interacción que ha habido entre ellos: si el usuario hace favorito un tweet, la puntuación asignada es de 1, y si le hace retweet, la puntuación es de 2. Además, puesto que es posible que un mismo usuario realice múltiples acciones sobre un mismo tweet, se considera que el peso del enlace que los une equivale a la suma de las interacciones existentes entre ellos.

Una vez se dispone de los centroides, el cálculo de la similitud entre los usuarios se realiza, al igual que en los casos anteriores, mediante la similitud por coseno.

$$f(u, v) = \frac{\sum_f u[f]\ v[f]}{\sqrt{\sum_f v[f]^2}}$$

3.4 Adaptación de algoritmos tradicionales de IR

En nuestro estudio del estado del arte no hemos encontrado en la literatura precedentes en la adaptación de algoritmos del campo de Recuperación de Información al campo de las redes sociales. Por ello, hemos escogido dos modelos probabilísticos utilizados típicamente en búsqueda para adaptarlos al ámbito objetivo y realizar una comparativa de su efectividad con los algoritmos de recomendación anteriores.

Tf-idf

Este método se utiliza típicamente en el ámbito de la búsqueda de documentos texto, ya que se basa en calcular qué tan relevante es una palabra para un documento en una colección (Baeza 1999). Consiste en el producto de dos partes: "tf" (*term frequency*), que depende de la frecuencia de un término en un documento, e "idf" (*inverse document frequency*), que es la frecuencia de aparición del término en la colección de documentos. A pesar de que "tf" aumenta proporcionalmente a la frecuencia del término en el documento, "idf" compensa con la frecuencia en la colección, de forma que se penalizan aquellos términos que son muy comunes en todos los documentos (ya que no discriminan bien entre distintos documentos), y favorecen aquellos que son menos comunes. A continuación se muestran las fórmulas de "tf" e "idf".

$$tf(t,d) = \begin{cases} 1 + \log frec(t,d), & si\ frec(t,d) > 0 \\ 0, & en\ otro\ caso \end{cases}$$

$$idf(t) = \log\left(1 + \frac{|D|}{1 + |D_t|}\right)$$

Donde $frec(t,d)$ representa el número de veces que la palabra t aparece en el documento d, D es el conjunto de todos los documentos de la colección, y D_t es el conjunto de documentos que contienen la palabra t.

Como se puede observar, el valor de *tf-idf* depende de un término y un documento, de forma que hay que utilizar algún método para calcular la similitud entre un documento y la consulta. Para ello, al igual que en el caso de los vecinos próximos, se pueden utilizar diferentes métodos, pero se ha escogido nuevamente el método del coseno. La fórmula completa de similitud entre un documento y una consulta se muestra a continuación. Nótese que se omite el módulo de la consulta puesto que es constante para todos los documentos y eliminarlo no afecta al orden del ránking.

$$f(d,q) = \frac{\sum_{t \in d \cap q} tfidf(t,d)\ tfidf(t,q)}{\sqrt{\sum_{t \in d} tfidf(t,d)^2}}$$

La adaptación del esquema *tf-idf* a la recomendación de contactos no es inmediata pues se concibió para la búsqueda de documentos mediante consultas, mientras que nuestro objetivo es recomendar contactos (en lugar de documentos) en ausencia de consulta, ni tan siquiera de un espacio de palabras. Es necesario por tanto reinterpretar el modelo para aplicarlo a nuestro problema. Para tal fin, los usuarios van a jugar el triple papel de documento, consulta y palabra. La frecuencia término-documento se puede entender como el peso del enlace existente entre dos usuarios; la frecuencia término-colección, como el tamaño del vecindario de un usuario; y el tamaño de la colección, como el número total de usuarios. Aplicando similitud por coseno de la misma forma que en la fórmula original, se obtiene la función de ránking buscada. Todo ello resulta en la siguiente formulación:

$$tf(w,u) = \begin{cases} 1 + \log weight(u,w), & si\ weight(u,w) > 0 \\ 0, & en\ otro\ caso \end{cases}$$

$$idf(w) = \log\left(1 + \frac{|U|}{1 + g(w)}\right)$$

$$tfidf(w,u) = tf(w,u)\ idf(w)$$

$$f(u,v) = \frac{\sum_{w \in N(u) \cap N(v)} tfidf(w,u)\ tfidf(w,v)}{\sqrt{\sum_{w \in N(v)} tfidf(w,v)^2}}$$

Donde $weight(u,w)$ es el peso del enlace existente del usuario *u* al usuario *w*, $|U|$ es el número de usuarios existentes en la muestra tomada y $g(w)$ es el grado del usuario *w*.

Modelos de lenguaje: query likelihood

Los llamados modelos de lenguaje de recuperación de información modelan los documentos y las consultas como eventos probabilísticos en espacios de probabilidad donde las variables aleatorias son las palabras. Una de las formulaciones más efectivas en este ámbito es el modelo llamado *query likelihood*, que genera el ránking de documentos en respuesta a una consulta estimando la probabilidad de que la consulta haya sido generada por la misma distribución que cada uno de los documentos (Ponte-Croft 1998). El desarrollo de este esquema sólo se fija en la frecuencia de los términos en cada documento, pero se realiza un suavizado (por ejemplo, el propuesto por Jelinek-Mercer) para que los términos que no aparezcan en el documento no tengan probabilidad nula. La formulación original del modelo es la siguiente:

$$f(d,q) = p(q|d) \propto \prod_{t \in q} p(t|d)$$

$$p(t|d) \sim (1 - \lambda)\frac{frec(t,d)}{|d|} + \lambda\, p_c(t)$$

$$p_c(t) = \frac{\sum_{d \in D} frec(t,d)}{\sum_{d \in D} |d|}$$

Donde $p(q|d)$ es la función de ranking que estima la probabilidad de que la consulta haya sido generada por la misma distribución que la del documento, $p(t|d)$ es la probabilidad de que un término de la query haya sido generado por la misma distribución que el documento y se estima por máxima verosimilitud con suavizado, λ es el parámetro de suavizado, $frec(t,d)$ es la frecuencia del término *t* en el documento *d*, $|d|$ es el tamaño del documento, y D es el conjunto de documentos existentes en la colección.

Para adaptar este modelo al contexto objetivo, transformamos tanto los términos como los documentos en usuarios, lo que da lugar a la necesidad de reinterpretar ciertos puntos de la fórmula original. Por un lado, la frecuencia de un término en un documento se convierte en el valor correspondiente al peso del enlace entre esos dos usuarios (en caso de ser dirigido, es necesario establecer la dirección del enlace). Por otro lado, el tamaño de un documento se reinterpreta como la suma de los pesos de los enlaces de sus vecinos (de nuevo, estableciendo dirección en caso de ser un grafo dirigido). Finalmente, la formula replanteada queda de la siguiente forma:

$$f(u,v) = p(u|v) \propto \prod_{w \in N(u)} p(w|v) \propto \sum_{w \in N(u)} \log\big(p(w|v)\big)$$

$$p(w|v) \sim (1-\lambda)\frac{weight(w,v)}{|\mathbf{v}|} + \lambda\, p_c(w)$$

$$p_c(w) = \frac{\sum_{x\in N(w)} weight(x,w)}{\sum_{u\in U}|\mathbf{u}|}$$

Donde $|\mathbf{u}|$ se entiende como la suma de los pesos de los enlaces del usuario u, y que se calcula de la forma $|\mathbf{u}| = \sum_{v\in N(u)} weight(v,u)$.

BM25

Es uno de los modelos probabilísticos más populares, y uno de los más efectivos hasta la fecha. Nació a partir del modelo clásico BIR (*Binary Independent Retrieval*), pero no está basado en el principio de ránking por probabilidad de Bernouilli sino en el de Poisson (Robertson 1995). A diferencia de BIR, BM25 tiene en cuenta la frecuencia de los términos y el tamaño de los documentos introduciendo únicamente dos nuevos parámetros, b y k, cuyos valores necesitan ser optimizados y se encuentran entre $[0,1]$ y $[0,\infty)$ respectivamente.

$$f(q,d) = \sum_{t\in q} \frac{(k+1)\, frec(t,d)}{k\left(1-b+\frac{b\,|d|}{\text{avg}(|d'|)}\right) + frec(t,d)}\, RSJ(t)$$

$$RSJ(t) = \log\frac{|D| - |D_t| + 0.5}{|D_t| + 0.5}$$

Donde k y b son parámetros del algoritmo, $\text{avg}(|d'|)$ es el tamaño medio de los documentos de la colección, $|D|$ es el número total de documentos presentes en la colección, y $|D_t|$ es el número de documentos de la colección que contienen el término *t*.

En este caso, al igual que en el caso anterior, para adaptar la fórmula a nuestro problema transformamos los términos y documentos en usuarios del grafo, la frecuencia de un término en un documento en el peso entre los dos respectivos usuarios, y el tamaño de un documento en la suma de los pesos de los enlaces del vecindario del usuario. Además, se interpreta el número de documentos de la colección como el número total de usuarios en el grafo, y el número de documentos que contienen un término, como el tamaño del vecindario del usuario correspondiente. Así, la fórmula final utilizada en este estudio es la siguiente:

$$f(u,v) = \sum_{w\in N(u)} \frac{(k+1)\, weight(w,v)}{k\left(1-b+\frac{b|\mathbf{v}|}{\text{avg}(|\mathbf{v}'|)}\right) + weight(w,v)}\, RSJ'(w)$$

$$RSJ'(w) = \log\frac{|U| - g(w) + 0.5}{g(w) + 0.5}$$

Como se verá más adelante en las comparativas empíricas, el valor óptimo del algoritmo se obtiene en el límite tomando el parámetro k tendiendo a infinito, por lo que se ha introducido tal variación a este modelo. Con esta configuración la función BM25 resulta en la siguiente expresión:

$$\lim_{k\to\infty} f(u,v) = \sum_{w\in N(u)} \frac{weight(w,v)}{1-b+\frac{b|\mathbf{v}|}{\text{avg}(|\mathbf{v}'|)}}\, RSJ'(w)$$

4. Nuevas dimensiones de utilidad de la red social

La efectividad de una recomendación se ha medido tradicionalmente con métricas orientadas a evaluar la calidad de ránking en términos de acierto con los gustos del usuario, y de forma individual, es decir, calculando un valor de la métrica por cada usuario y haciendo la media de los resultados. En este estudio se amplía por un lado la perspectiva tradicional considerando otras dimensiones de calidad de recomendación como la novedad y la diversidad de la misma. Además, planteamos un nuevo enfoque orientado a valorar la ganancia global en toda la red y no de forma aislada para cada usuario.

A continuación se describirán los diferentes puntos de vista de evaluación de la calidad de recomendación, detallando las métricas utilizadas en el presente trabajo para realizar la comparación entre los diferentes sistemas de recomendación.

4.1 Acierto

Este tipo de métricas tienen como objetivo medir el grado en que se satisfacen los gustos del usuario objetivo al realizar una recomendación. Un contacto recomendado se considera que ha sido relevante para el usuario objetivo cuando éste considera que se trata de una recomendación que le ha resultado útil.

A este grupo pertenecen las métricas siguientes del campo de la Recuperación de Información, que adaptamos a nuestro contexto de recomendación de contactos.

- **Cobertura:** número de usuarios a los que el algoritmo ha sido capaz de realizar una recomendación (Baeza 2011). En los casos normales, esta métrica estará siempre es 1 o casi (es decir, cobertura total), pero conviene monitorizar esta métrica ya que ciertos algoritmos presentan deficiencias de cobertura, pudiendo distorsionar el valor de otras métricas (por ejemplo, si sólo se recomienda a los usuarios más "fáciles", puede salir una precisión muy alta que no es la que realmente le corresponde al algoritmo en global).

$$Cobertura = \frac{\{u|\, R(u) \neq \emptyset\}}{|U|}$$

- **Precisión**: fracción de contactos recomendados que han resultado ser relevantes para el usuario objetivo (Baeza 2011). Representa la tarea de que un usuario quiere encontrar un número razonable de contactos relevantes y valora el ratio de contactos relevantes por unidad de esfuerzo que supone examinar las recomendaciones.

$$P(u) = \frac{|\{v|v \in R(u), (u,v) \in E_{test}\}|}{|R(u)|}$$

- **Recall**: similar a Precisión, representa la fracción de contactos relevantes para el usuario que han sido incluidos en la recomendación al mismo (Baeza 2011). Esta métrica representa una tarea donde el usuario quiere encontrar todos los contactos relevantes y no repara en el esfuerzo de examinar los que sean irrelevantes.

$$R(u) = \frac{|\{v|v \in R(u), (u,v) \in E_{test}\}|}{|\{v|(u,v) \in E_{test}\}|}$$

- **nDCG (*Discounted Cumulative Gain*):** mide la relevancia de los contactos teniendo en cuenta su posición en el ránking, de forma que un valor alto de nDCG indica que hay contactos relevantes en las primeras posiciones. Así, esta métrica puede valorar más un contacto poco relevante en las primeras posiciones que uno con mayor relevancia que esté más abajo en el ránking. Admite además la distinción de diferentes grados de relevancia (a diferencia de métricas como precisión y recall, que consideran relevancia binaria) (Järvelin 2000).

$$nDCG(u) = \frac{DCG(u)}{IDCG(u)}$$

$$IDCG(u) = max\, DCG(u)$$

$$DCG(u) = \sum_{v_k \in R(u)} \frac{g'(u, v_k)}{log_2(k+1)}$$

$$g'(u,v) = 2^{weight(u,v)+1} - 1$$

Donde *k* es la posición del ítem en el ránking, g es el grado de relevancia, e IDCG es el valor DCG del mejor ránking posible.

DCG suma grados de relevancia penalizados por lo tarde que aparezcan en el ránking, y la normalización por IDCG permite neutralizar la diferencia entre usuarios más fáciles o difíciles de satisfacer (es decir evitar que la métrica dependa del número grande o pequeño de contactos relevantes que tenga el usuario). El grado de relevancia se obtiene considerando como juicio de relevancia los enlaces de la partición de test, de forma que el peso de estos enlaces se utilizan para realizar un mapeo de valores según diferentes variantes (lineal, binaria, logarítmica, etc.). En este caso, se ha decidido utilizar el mapeo exponencial.

Las métricas de relevancia son importantes porque el objetivo principal es que al usuario le gusten las recomendaciones que se le hacen. Sin embargo, es interesante para éste que no sólo se acierten sus gustos: por ejemplo, si a un usuario le gustan mucho las películas de acción, se le recomendarán más películas de acción que no haya visto. El inconveniente es que con ello se encasilla al usuario, mostrándole siempre películas del mismo género. Para que esto no suceda, se motiva la evaluación de otras dimensiones de calidad además de la relevancia, como la novedad y la diversidad. Éstas métricas permiten comparar los diferentes algoritmos para ver en qué medida hacen que el usuario pueda ampliar sus gustos y añadir variedad a su experiencia.

4.2 Novedad

En esta línea de métricas se valora la cantidad de novedad que se le aporta a un usuario objetivo en la recomendación que se le realiza. En otras palabras, se mide qué tan diferente son los contactos recomendados frente a los que han sido observados por el usuario anteriormente.

Las métricas que hemos adaptado al contexto de las redes sociales incluyen las siguientes:

- **EPC (*Expected Popularity Complement*):** De forma intuitiva, representa la probabilidad promedio de que un usuario no conozca los contactos que se le han recomendado (Vargas 2011).

$$EPC(u) = \frac{1}{|R(u)|} \sum_{v \in R(u)} \big(1 - p(known|v)\big)$$

$$p(known|v) \sim \frac{g_{in}(v)}{|U|}$$

- **EPD (*Expected Profile Distance*):** Se basa en el cálculo de la distancia media entre los contactos recomendados y los que ya tienen enlace con el usuario, de forma que su cercanía o lejanía son indicadores de novedad para dicho usuario (Vargas 2011). La similitud entre dos contactos se calcula normalmente con un enfoque basado en contenido, ya que la métrica supone que hay un espacio de características de los usuarios.

$$EPD(u) = \frac{1}{|U||R(u)|} \sum_{u \in U} \sum_{v \in R(u)} dist(v,u)$$

$$dist(v,u) = \frac{1}{g_{out}(u)} \sum_{w:(u,w) \in E_{train}} dist(v,w)$$

$$dist(v,w) = 1 - sim(v,w)$$

Estas métricas se basan únicamente en el aporte de novedad intrínseco a los contactos del ránking. Existen otras versiones que tienen en cuenta la posición en el ránking y la relevancia de los contactos (Ricci 2011), pero para este trabajo optamos por esta versión más sencilla y estándar.

4.3 Diversidad

Además de la novedad de la recomendación que se realiza al usuario, otro punto interesante de vista es que ésta sea variada. Las métricas de diversidad se orientan a captar esta cualidad, es decir, en qué medida los contactos recomendados son diferentes entre sí.

En la métrica de novedad de EPD se habló de similitud entre contactos según su contenido, y esto mismo se va a tratar aún con más importancia con los algoritmos de diversidad. En este caso, cada contacto tiene asociado un conjunto de *subtopics* (o "aspectos" en este contexto), de forma que los contactos pueden agruparse según los aspectos que tengan en común.

A partir de este punto, la elección de los aspectos a tratar abre numerosos y diferentes espacios de diversidad. En este estudio, principalmente será objeto de análisis la diversidad basada en hashtags (exclusivo de la red de Twitter) y de la diversidad basada en comunidades obtenidas mediante algoritmos de modularidad, siguiendo el concepto de enlaces fuertes y débiles (De Meo 2014).

Las métricas que se van a aplicar se basan en las propuestas documentadas en (Clarke 2008, Vargas 2011). A continuación se resumen las métricas de este grupo que se han adaptado del campo de Recuperación de Información para ser utilizadas en la comparativa de este estudio:

- **ERR-IA (*Intent-Aware Expected Reciprocal Rank*):** Se trata de una versión más compleja de MRR-IA en la además de la posición de los contactos recomendados, se tiene en cuenta su grado de relevancia. Con esto se para comprobar que aquellos más relevantes estén en las posiciones más altas del ránking, pero al mismo tiempo penalizando la repetición (redundancia) de subtopics que ya han aparecido en posiciones anteriores a cada contacto del ránking. Nuevamente, esto se realiza para cada uno de los aspectos de forma independiente, realizando una media final (Chapelle 2011).

$$ERR\text{-}IA(u) = \sum_{z \in Z} ERR(u,z)\, p(z|q)$$

$$ERR(u,z) = \sum_{v_k \in R(u)} \frac{1}{k} p(r|v_k) \prod_{j<k} (1 - p(rel|v_j))$$

$$p(r|v) \sim \frac{2^{g(v)} - 1}{2^{g_{max}}}$$

  Donde $p(z|q)$ se suele tomar uniforme $p(z|q) \sim 1/|Z|$ (siendo Z el conjunto de todos los aspectos posibles) a falta de información particular sobre qué aspectos son más o menos probables para un usuario.

- **S-Recall (*Subtopic Recall*):** Representa el porcentaje de aspectos cubiertos por al menos uno de los contactos del ránking. En este caso, sólo los contactos que han resultado ser relevantes para el usuario son tenidos en cuenta (Zhai 2003).

$$S\text{-}Recall(u) = \frac{1}{|Z|} \left| \bigcup_{\substack{v \in R(u) \\ (u,v) \in E_{test}}} subtopics(v) \right|$$

- **Gini:** Tomada del campo de la ecología y la economía, esta métrica se ha aplicado a la recomendación para medir la desigualdad entre la cantidad de veces que cada contacto ha sido recomendado (Shani 2011, Castells 2015). Puede tomar valores comprendidos en el rango $[0,1]$, siendo cero cuando todos los contactos son recomendados el mismo número de veces, y uno cuando se recomienda exclusivamente el mismo contacto a todos los usuarios.

$$G = \frac{1}{|Z| - 1} \sum_{j=1}^{|Z|} (2j - |Z| - 1) p(v_j)$$

$$p(v) = \frac{|\{u | v \in R(u)\}|}{\sum_u |R(u)|}$$

  Donde $p(v)$ representa la fracción de usuarios a los que el contacto v_j ha sido recomendado, y la suma se realiza ordenando de forma ascendente según los valores de $p(v)$.

Las métricas anteriormente descritas (acierto, novedad y diversidad) se fijan en la calidad desde diferentes puntos de vista de la recomendación a cada uno de los usuarios de forma individual. Sin embargo, aquí planteamos que tiene sentido considerar una evaluación global, mirando cómo cambia la red y sus propiedades con las recomendaciones realizadas a todos los usuarios en conjunto. De este planteamiento surge la idea de inclusión al estudio las métricas de diversidad estructural, las cuales se detallan a continuación.

4.4 Propiedades globales de red

Tal y como se ha dicho anteriormente, en la literatura la forma más común y extendida de evaluar los resultados de los algoritmos de recomendación ha sido orientada a la ganancia individual del usuario. Existe una gran cantidad de métricas, pero en casi todas ellas la ganancia global (ya sea en términos de acierto, novedad, diversidad, etc.) se define como el promedio de las ganancias individuales (Gini y Cobertura son excepciones a esta norma).

Una de las partes más innovadoras de este trabajo es el estudio de la idea de enfrentar este planteamiento de ganancia individual de los usuarios con el de ganancia global de la red. Proponemos que en lugar de recomendar al usuario mirando sólo a éste, se valore las propiedades de la red, de forma que se modifique ligeramente las recomendaciones para que la red pueda mejorar, por ejemplo, en cuanto a la propagación de información en la misma.

Para particularizar esta idea, hemos tomado métricas tradicionales que estudian las características del grafo que forma la red de usuarios. El área de las redes sociales ha dedicado un esfuerzo importante en definir métricas que capturen propiedades globales relevantes de las redes, y es otra idea original de este trabajo explorar la posibilidad de adaptarlas para los fines planteados.

Se ha considerado como objeto principal de interés la modularidad del grafo, ya que la separación por comunidades que se realiza al calcularla puede utilizarse como nueva dimensión de diversidad. Además, esta línea lleva a conectar varios conceptos: comunidad, diversidad, enlaces débiles y métricas de redes.

Para este estudio, hemos seleccionado aquellas métricas de diversidad estructural cuya la noción que las motiva pueda tener conexión con la diversidad, novedad u otras de bien común. De esta forma, haremos experimentos para ver qué métricas parecen indicar cosas de interés.

Las métricas finalmente implementadas son las siguientes:

- **Coeficiente de clustering global:** Refleja en qué medida el grafo está formado por enlaces redundantes, es decir, en qué medida los usuarios están relacionados por mediación de otros usuarios (Wasserman 1994). Así, se puede entender que un grafo será más diverso cuanto más bajo sea su valor de coeficiente de clustering global.

$$CCG(G) = \frac{|\{(w,v)|(w,v),(v,u),(u,w) \in E_{train}\}|}{|\{(w,v)|(v,u),(u,w) \in E_{train}\}|}$$

- **Coeficiente de clustering promedio:** Se trata de una definición alternativa a la anterior donde en lugar de realizar el cálculo sobre todo el grafo completo, se realiza parcialmente para cada nodo y se hace finalmente la media (Watts 1998).

$$CCLA(G) = \frac{1}{|U|}\sum_{u\in U}\frac{|\{(w,v)|(w,v),(v,u),(u,w)\in E_{train}\}|}{g_{out}(u)\ g_{in}(u) - |\{(w,v)|(v,u),(u,w)\in E_{train}\}|}$$

  Es menos costosa que el coeficiente de clustering global, pero es común que predominen los nodos de grado bajo y al hacer la media, los resultados se vean trastocados.

- **Modularidad de grado:** Con esta métrica se mide si los usuarios con alto grado tienen muchos enlaces entre ellos pero pocos con los de grado bajo, y viceversa. En este caso, la fórmula cambia levemente según si se trata de un grafo dirigido o uno no dirigido.

$$DM = \frac{|E_{train}|\sum_{(u,v)\in E_{train}} g_{out}(u)\ g_{out}(v) - (\sum_{u\in U} g_{out}^2(u))^2}{|E_{train}|\sum_{u\in U} g_{out}^3(u) - (\sum_{u\in U} g_{out}^2(u))^2}$$

$$UM = \frac{4|E_{train}|\sum_{(u,v)\in E_{train}} g(u)\ g(v) - (\sum_{u\in U} g^2(u))^2}{2|E_{train}|\sum_{u\in U} g^3(u) - (\sum_{u\in U} g^2(u))^2}$$

- **Número de enlaces "débiles" (*Edges*):** Cuenta el número de arcos débiles, que son aquellos que conectan dos usuarios pertenecientes a dos comunidades diferentes del grafo (De Meo 2014).

$$Edges = |\{(u,v)|(u,v)\in E_{train}, comm(u)\neq comm(v)\}|$$

- **Puentes (*Bridges*):** Granovetter define puente como un arco (u,v) que proporciona el único camino entre *u* y *v* (Granovetter 1973). Debido a que las redes sociales suelen tener una única componente gigante fuertemente conexa y el resto del grafo es muy poco significativo, no es clara la utilidad que aportaría la conexión con estos nodos residuales. Por ello, en este contexto se utiliza otra noción más relajada propuesta por De Meo et al. (De Meo 2014) basada en comunidades y no en componentes conexas. La métrica creada a raíz de esta noción cuenta el número de enlaces que atraviesan dos comunidades y además cierran la conexión entre ellas. Esta métrica permite identificar los enlaces que son clave en el paso de información, de forma que si se quita un puente, la información de una comunidad puede no llegar a la otra (puede haber otro puente que conecte las mismas comunidades).

$$B = |\{(u,v)|(u,v)\in E_{train}, comm(u)\neq comm(v)\wedge comm(u)\in succ(comm(v))\}|$$

  Donde $comm(u)$ es la comunidad a la que pertenece el usuario u, y $succ(c)$, el conjunto de comunidades a la que es accesible desde la comunidad c.

- **Arraigo (*Embeddedness*):** El arraigo mide si cada enlace conecta usuarios con muchos contactos en común, de forma que se tiene una idea de la probabilidad de que el par de usuarios pertenezca al mismo grupo al que pertenecen sus contactos comunes (Moody 2003). Para calcularlo, se realiza la media entre el valor de arraigo de cada enlace, de forma que se estima cuánta redundancia hay en el grafo de forma global. Por ello, al igual que el coeficiente de clustering, para que un grafo sea diverso el valor del arraigo debe ser bajo.

$$A = \frac{1}{|E_{train}|} \sum_{(u,v) \in E_{train}} A'(u,v)$$

$$A'(u,v) = \frac{|\{w|(u,v),(u,w),(w,v) \in E_{train}\}|}{g_{out}(u) + g_{in}(v) - 2 - |\{w|(u,v),(u,w),(w,v) \in E_{train}\}|}$$

- **Arraigo cero:** Utiliza la fórmula anterior para contar el número de pares de usuarios conectados que tienen arraigo cero, de forma que se obtiene la cantidad de enlaces que no conectan usuarios del mismo grupo. Así, cuanto mayor sea el arraigo cero de un grafo, más diverso será.

$$A_0 = |\{(u,v)|(u,v) \in E_{train}, A'(u,v) = 0\}|$$

- **Modularidad:** Mide qué tan fuerte es la división en comunidades de un grafo (Newman 2004), de forma que un valor alto significa que hay muchos enlaces entre los usuarios de la misma comunidad pero pocos con los usuarios de otras comunidades.

$$Q = \frac{1}{2m} \sum_{ij} \left[w_{ij} - \frac{g(u_i)\, g(u_j)}{2m} \right] \delta\left(comm(u_i), comm(u_j) \right)$$

  Donde m es la suma de los pesos de los enlaces del grafo, w_{ij} es el elemento de la matriz de adyacencia que indica el número de enlaces entre los nodos u_i y u_j y δ es la función delta de Kronecker (vale 1 si $comm(u_i) = comm(u_j)$ y 0 en caso contrario).

4.5 *Rerankers* para la optimización de métricas

Establecer un criterio de evaluación determina también el objetivo que pueden perseguir los algoritmos para maximizarlo. Es posible de hecho definir un algoritmo de recomendación consistente en un método de maximización de una métrica de evaluación dada. Sin embargo, es común que la tarea de recomendación requiera optimizar más de un criterio (por ejemplo, acierto y diversidad), por lo que una estrategia común consiste en partir de un buen algoritmo de recomendación que alcance un nivel de acierto, y a la salida de este algoritmo aplicarle una reordenación moderada que mejore otra métrica (por ejemplo, novedad o diversidad) en las primeras posiciones del ránking. Siguiendo esta línea, en nuestro estudio incluiremos algunos algoritmos de *reranking* para, a partir de recomendaciones generadas para una ganancia individual de los usuarios, orientarlas a la búsqueda la optimización de ciertas características de la red.

Para nuestro estudio, seleccionamos como objetivo de optimización dos características de la red: la modularidad y la diversidad. En todos los casos, para cada usuario objetivo se genera un nuevo ranking con los usuarios recomendados originalmente según una función $rerank(u,v)$ que cada algoritmo de reranking define. Para obtener la nueva puntuación, se realiza una ponderación mediante combinación lineal entre las dos partes según un parámetro λ, la posición en el ránking original ($old(u,v)$) y en el nuevo ($new(u,v)$).

De esta forma, las fórmulas utilizadas son las siguientes:

$$\bar{f}(u,v) = (1-\lambda)\, old(u,v) + \lambda\, new(u,v)$$

$$old(u,v) = \frac{|R(u)| - rank(u,v) + 1}{|R(u)|}$$

$$new(u,v) = \frac{\sum_{k=1}^{rep} |R(u)| - (k+n) + 1}{rep\ |R(u)|} = \frac{|R(u)| + 1 - n - \frac{rep+1}{2}}{|R(u)|}$$

Donde $|R(u)|$ es el número de contactos recomendados al usuario objetivo u, $rank(u,v)$ es la posición del contacto recomendado v en el ranking del usuario objetivo u, rep es el número de contactos que han obtenido la misma puntuación en la función de $rerank(u,v)$ que el contacto objetivo, y n es la primera posición en el nuevo ránking donde el score coincide con el del contacto objetivo.

Para realizar la ponderación correctamente, es necesario normalizar cada una de las partes implicadas en la combinación. Entre las diferentes opciones documentadas en la literatura, se ha escogido para aplicar en este caso la normalización *rank-sim*, que ha demostrado buen funcionamiento (Lee 1997).

Como se puede observar, en la función de $new(u,v)$ se le otorga la misma puntuación a aquellos contactos que hayan recibido un nuevo score igual, a pesar de que la posición en el ranking nuevo sea diferente. Con esto se pretende eliminar sesgos al hacer la ponderación según las posiciones en el nuevo ránking.

Los rerankers desarrollados para el presente estudio entonces son los siguientes.

- **Infomap:** Maximiza la modularidad calculada por el algoritmo de Infomap. La fórmula que utiliza para calcular el nuevo score da resultados binarios, dependiendo si el nuevo enlace a introducir conecta dos nodos de diferentes comunidades o no.

$$rerank(u,v) = \begin{cases} 1, & si\ comm(u) \neq comm(v) \\ 0, & en\ caso\ contrario \end{cases}$$

- **Comm-Gini:** Busca que se recomienden de la forma más uniforme posible todas las comunidades a los usuarios objetivo (no de forma particular para cada usuario sino de forma conjunta para todos los usuarios). El reranking entonces se realiza estableciendo la puntuación de cada contacto según el número de veces que su comunidad ha sido recomendada y actualizando los valores ante cada cambio.

$$rerank(u,v) = \frac{|R(u)||U_{test}| - |\{w|w \in R(u), comm(w) = comm(v)\}|}{|R(u)|\ |U_{test}|}$$

- **XQuAD (*eXplicit Query Aspect Diversification*):** Se trata de la adaptación de un algoritmo de diversificación de resultados de búsqueda en el campo de la Recuperación de Información, concretamente, el diversificador más efectivo en esa área (Santos 2010). La función objetivo de este método tiene un fundamento probabilístico, y persigue maximizar la variedad de aspectos cubiertos lo antes posible en el ránking:

$$rerank(u,v_k) = \sum_h p(h|u)p(v_k|u,h)\prod_{j=1}^{k-1}\left(1 - p\left(v_j|u,h\right)\right)$$

Donde v_k representa el contacto k-ésimo en el ránking de recomendación, h recorre el conjunto de aspectos, $p(h|u)$ representa cuán relevante es el aspecto h para

el usuario objetivo, y $p(v|u,h)$ representa cómo de relevante es el usuario v para el aspecto h. Esta función objetivo favorece los contactos más relacionados con los aspectos más relevantes para el usuario objetivo, a la vez que mediante el producto en j penaliza los contactos v_k más redundantes en cuanto a que cubren aspectos ya cubierto por contactos v_j que les preceden en el ránking de recomendación.

En todos estos casos, los algoritmos de reranking han buscado la optimización de la diversidad o de la modularidad de la red de forma pura, es decir, buscando maximizar únicamente una característica. Sin embargo, para el presente trabajo se ha desarrollado un nuevo reranker que busca optimizar tanto la modularidad como la diversidad de las recomendaciones, y se detalla a continuación.

- **Infomap-Gini:** Realiza un reordenamiento progresivo del ránking de cada usuario subiendo al top n de contactos a recomendarle aquellos que cumplen dos condiciones: si pertenece a una comunidad diferente a la del usuario objetivo (igual que con el algoritmo de reranking de Infomap), y además si el cambio en el ranking realiza una mejora en el valor Gini global (no por comunidad sino por contacto). En este caso no se utiliza una fórmula $rerank(u,v)$, ya que no se crea un nuevo score para los contactos recomendados sino que se realizan los cambios en el ordenamiento de forma directa según las condiciones anteriormente descritas.

4.6 Diversidad en el flujo de información

Los algoritmos de reranking anteriores buscan la diversificación estructural, fomentando en general que el usuario reciba en su recomendación contactos pertenecientes a otras comunidades. Sin embargo, ¿qué es lo que aporta una red con diversificación estructural, es decir, menos modular? Es difícil dar respuesta tanto teórica como empíricamente a esta pregunta, pero planteamos la hipótesis de que una red más diversa fomenta un flujo de información más diverso. A pesar de que a primera vista puede parecer una idea evidente, no lo es: mientras que la diversidad estructural se define según la densidad local de enlaces entre los usuarios (comunidades), la diversidad de información se define por el contenido que cada usuario recibe (por ejemplo, tweets y hashtags en la red social de Twitter).

Para el estudio de esta hipótesis, se ha desarrollado un programa donde se simula el flujo de información entre usuarios pertenecientes a una muestra de la red social de Twitter. De esta forma, se puede observar y comparar cómo influyen las recomendaciones realizadas por cada uno de los rerankers anteriores en la forma de propagar la información a través de los usuarios si éstos aceptasen las recomendaciones que reciben. El elemento principal en el que nos vamos a fijar es en la diversidad en la información propagada, es decir, qué tan diversa es la información que pasa por cada uno de los usuarios.

Mediante las simulaciones se pretende entonces comparar el comportamiento de la red original y la red que resultaría de diferentes formas de recomendar, simulando el flujo de tweets a través de estas versiones de la red. Los tweets seleccionados serán aquellos que los propios usuarios escribieron en Twitter, y se simularán retweets mediante un procedimiento aleatorio que emula patrones de selección y redifusión (retweet) de tweets.

Como describiremos en la sección xxx, el método consistirá en partir de la red de usuarios original con las conexiones existentes entre ellos, y añadir como nuevos enlaces las $top\ k$ recomendaciones realizadas a cada usuario por el algoritmo de recomendación que se va a analizar. Para cada usuario se seleccionarán los n más recientes que contengan algún hashtag.

Procedimiento de simulación

Como se ha dicho anteriormente, la simulación se realiza por iteraciones. En cada iteración, cada usuario se aborda de forma individual de la siguiente forma. Si el usuario no tiene ningún tweet disponible para propagar, o no tiene ningún follower al que propagarle la información, se pasa al siguiente usuario. Si no, se realiza una selección aleatoria de tweets según el ratio definido r, y cada uno de estos se propaga o no a sus followers según una probabilidad p. La simulación finaliza cuando no se puede propagar más información, es decir, que ningún usuario tiene ningún tweet para retweetear o sí tiene pero no tiene followers a quien pasárselo.

Cada usuario cuenta con varios conjuntos de tweets diferenciados donde almacena los diferentes tweets con los que tiene alguna relación:

- Iniciales: son sus tweets, nadie más los tiene y son "etiquetados" con la comunidad del usuario.
- Disponibles: sus tweets más los que ha recibido de sus followees, sin considerar el número de veces que se ha recibido el tweet.
- Retwitteados: los tweets que el usuario ha retwitteado. Esto se guarda para evitar propagar más de una vez un mismo tweet.
- Recibidos: los tweets que va recibiendo el usuario de sus followees en la iteración actual se almacenan en este conjunto, hasta que al finalizar la iteración se pasan estos tweets al conjunto de tweets disponibles.

Métricas

En cada iteración se realiza una serie de mediciones, algunas por usuario y otras de forma global, de forma que se pueda medir el progreso en ciertos aspectos de la propagación de información en la red. En este caso, se han tenido en cuenta diferentes métricas para evaluar aspectos de diversidad, donde destacan principalmente las siguientes:

- **Hashtag-recall (HT-Recall):** Se trata del porcentaje de hashtags que han recibido los usuarios de media. Se calcula para cada iteración de la siguiente forma:

$$HT\text{-}Recall(G) = \frac{\sum_{u \in U} receivedHashtags(u)}{|U|\,|H|}$$

- **Community-recall (Comm-Recall):** Muy similar al anterior, mide el porcentaje de comunidades de las que ha recibido tweets. La fórmula utilizada es la siguiente:

$$Comm\text{-}Recall(G) = \frac{\sum_{u \in U} receivedCommunities(u)}{|U|\,|C|}$$

- **Hashtag-Gini (HT-Gini):** Es la media del valor de Gini respecto a los hashtags recibidos por cada usuario. Cuanto mayor sea este valor, más equilibrada será la cantidad de tweets que los usuarios reciban con los hashtags existentes.

- **Community-Gini (Comm-Gini):** Igualmente, es la media del valor de Gini esta vez respecto a las comunidades de las que se ha recibido algún tweet. Un valor alto implica que, de media, los usuarios reciben en torno a la misma cantidad de tweets de cada comunidad existente en la red.
- **Número de tweets (Tweets):** Número de tweets que recibe de media cada usuario en cada iteración. Se trata de una métrica muy simple, pero puede dar una idea de la cantidad de información que se propaga por la red, de forma que puede haber algoritmos que favorezcan mucho que toda la información llegue a la mayor cantidad de usuarios.

$$Tweets(G) = \sum_{u \in U} receivedTweetsCurrentIteration(u)$$

- **Community-Tweets (Comm-Tweets):** Representa la media del porcentaje de tweets que recibe un usuario de otros de comunidades diferentes a la suya. Con esto se pretende medir en qué medida los algoritmos consiguen que le llegue a los usuarios información de otras comunidades, es decir, en qué medida favorecen la diversidad en la propagación de información.

$$Comm\text{-}Tweets(G) = \frac{\sum_{u \in U} \frac{receivedTweetsFromOtherCommunities(u)}{receivedTweets(u)}}{|U|}$$

Es de destacar que las simulaciones tardan mucho en ejecutarse, por lo que además de optimizar al máximo el código, se ha serializado las clases pertinentes para que cada una hora de ejecución guarde el estado completo de la simulación, de forma que en caso de que se interrumpa, se pueda recuperar desde un punto cercano.

Como extra, se ha creado una variación de esta simulación en la que se realiza únicamente las primeras $nIterations$ iteraciones y se repite la ejecución un número de $nRepeats$ veces. Con esto, se consigue que los valores de las métricas obtenidos sean un promedio de todas las ejecuciones realizadas, obteniendo unos resultados algo más fiables que en el caso de la simulación completa.

5. Experimentos

De modo similar a otros campos relacionados como el aprendizaje automático o la recuperación de información, cabe distinguir dos tipos diferenciados de metodología experimental en el campo de los sistemas de recomendación: las pruebas offline y las pruebas online. En las pruebas offline, en el caso particular de la recomendación de contactos, se toma una muestra representativa de la red social objetivo y se divide en dos, entrenamiento y test, para probar sobre ella los diferentes algoritmos y evaluarlos sobre las dimensiones deseadas. Los experimentos offline tienen limitaciones tales como los posibles sesgos de la muestra de datos (y/u otras diferencias con las características de los datos en su versión completa), el sobreajuste, la imposibilidad de recoger reacciones directas y explícitas de los usuarios a la salida de los algoritmos, así como la imposibilidad de reproducir exactamente las condiciones de un contexto de ejecución real. Los experimentos online pueden superar estas limitaciones, pero debido a su coste (pues precisan la disponibilidad o reclutamiento expreso de usuarios ad-hoc) suelen presentar otras limitaciones en cuanto a la dimensión y escala (en número de usuarios, de casos, de sistemas, etc.) de las pruebas, así como la reproducibilidad del experimento. Cabe decir que por todo ello que ambas metodologías, online y offline, pueden ser un buen complemento la una de la otra.

En el presente trabajo se han analizado varios sistemas de recomendación evaluando sus resultados desde numerosos puntos de vista (pruebas offline), y se han seleccionado de ellos los más interesantes o significativos para comparar su eficiencia en el entorno de usuarios finales mediante una aplicación móvil (pruebas online). Además, se trata una de las muchas preguntas a investigar que surgen a raíz del contexto específico en el que nos encontramos en este estudio, que es la red social de Twitter.

Tras identificar primeramente las preguntas sobre las que los experimentos persiguen arrojar luz, en las secciones que siguen se explican en detalle los experimentos llevados a cabo en ambas líneas, detallando la configuración, el contexto y los resultados obtenidos.

5.1 Preguntas de investigación

La investigación abordada en este trabajo, y en particular los experimentamos que documentamos en este capítulo, abordan varias preguntas de investigación entre las que se incluyen las siguientes:

RQ1. ¿Qué algoritmos de recomendación de contactos pueden ser más efectivos en Twitter?

RQ2. ¿Es posible obtener algoritmos de recomendación efectivos adaptando modelos de Recuperación de Información? ¿Y algoritmos de recomendación de ítems?

RQ3. De cara a la recomendación de contactos en una red social asimétrica, ¿Qué representa mejor los intereses de los usuarios: los usuarios a quienes siguen, o los usuarios por los que son seguidos?

RQ4. ¿Qué significa diversidad en una red social? ¿Y en el contexto de la recomendación?

RQ5. ¿Tiene sentido adaptar a la recomendación de contactos métricas de novedad y diversidad de recuperación de información y sistemas de recomendación?

RQ6. ¿Qué métricas globales de red pueden aportar un criterio relevante en la evaluación de un sistema de recomendación de contactos? ¿Puede ser útil recomendar enlaces débiles?

RQ7. ¿Qué métricas candidatas a captar propiedades relevantes de la recomendación son realmente distintas o hasta cierto punto equivalentes entre sí?

RQ8. ¿Puede utilizarse la modularidad como medida de diversidad estructural, y es una métrica estable frente a diferentes algoritmos de detección de comunidades que la aproximen?

RQ9. ¿Qué efectos puede tener la diversidad estructural promovida por un algoritmo de recomendación en la diversidad del flujo de información a través de la red?

Hemos llevado a cabo en este trabajo una amplia labor experimental orientada a abordar estas preguntas, cuyo desarrollo, resultados y observaciones pasamos a describir a continuación.

5.2 Pruebas offline

Comenzamos por detallar la configuración de los experimentos realizados, explicando tanto la forma de obtener el muestreo de la red social objetivo como el entorno de trabajo, los espacios de atributos utilizados y las librerías incluidas en el proyecto realizado. A continuación, mostraremos y analizaremos los resultados obtenidos, centrándonos en algunos de los barridos de parámetros más relevantes, el estudio de la dirección de los arcos para seleccionar el mejor vecindario, análisis y comparación de los rerankers, y la diversidad en la propagación de información a la que puede dar lugar la diversidad estructural que proporcionan las recomendaciones.

5.2.1 Configuración experimental

Resumimos a continuación los detalles de la puesta a punto experimental: el conjunto de datos utilizado, el entorno de trabajo donde se han realizado las pruebas, los espacios de atributos utilizados, y las librerías públicas utilizadas.

Conjunto de datos

Para el presente trabajo se ha tomado como objetivo de estudio principal la red social de Twitter[8]. De forma resumida, se trata de una plataforma de *microblogging* (publicación de contenido limitado a 140 caracteres, los llamados tweets) que contiene una red asimétrica de usuarios. Esta red consiste en enlaces de tipo "*follow*", de forma que si un usuario "sigue" a otro, el primero recibe en su *timeline* las publicaciones que realice el segundo. Así, el primero es "seguidor" o follower del segundo y el segundo es followee del primero. Como extra, esta plataforma permite marcar tweets como favoritos y/o hacerles retweet (re publicación del tweet de otro usuario).

Existe a disposición pública la API oficial de Twitter (API REST) donde se pueden realizar peticiones para obtener todo tipo de información existente en la plataforma (usuarios, tweets, seguidores, etc.). Sin embargo, la recopilación de datos es lenta, ya

[8] Twitter, https://twitter.com/

que el uso de la API está sujeto a una limitación del número de peticiones dependiendo del tipo de petición (15-450 peticiones en una ventana de 15 minutos). Además, es necesario disponer de una cuenta autorizada para ello, que debe estar formada por una cuenta de Twitter que tenga un número de teléfono único y comprobado.

En cuanto a la forma de realizar la partición de entrenamiento y test, es necesario realizarlo con precaución ya que es muy probable que se den problemas relacionados principalmente con aciertos triviales. Por ejemplo, si en los datos de entrenamiento el usuario *u* ha hecho retweet a otro usuario *v*, y entre los datos de test se tiene que *u* sigue a *v*, los algoritmos tenderán a recomendar que *u* siga a *v* y se realizaría un acierto trivial. Para evitar estos problemas y obtener una división limpia entre el entrenamiento y el test, nos hemos basado en una partición temporal entre entrenamiento y test que refleje un caso real lo mejor posible. En el caso de la red social de Twitter, no toda la información tiene una marca temporal asociada (por ejemplo, la creación de los enlaces follow), por lo que no es inmediato este tipo de partición.

Para el presente trabajo, se ha tomado como partición de entrenamiento el muestreo completo utilizado en (Alhambra 2014), y se ha realizado un muestreo nuevo para el test, tomando los datos nuevos de los mismos usuarios a partir de la fecha del muestreo de entrenamiento. De esta forma, se obtiene una partición completamente limpia (sin solapamientos entre entrenamiento y test) y de forma temporal.

Este nuevo conjunto de test se ha obtenido mediante el uso de la API de Twitter, obteniendo los tweets realizados y las interacciones entre los usuarios de la muestra realizados tras la fecha de finalización de la partición de entrenamiento. En cuanto a los nuevos enlaces follow, a pesar de que no tienen asociada una marca temporal, la lista de followees devuelta por API sí está en orden temporal. De esta forma, para reconocer los enlaces nuevos hemos vuelto a recoger los followees de cada usuario para seleccionar aquellos que no estaban en entrenamiento, que son por tanto posteriores en el tiempo, y los usamos como datos de test. Una vez se conocen los nuevos enlaces, basta con agregar aquellos enlaces a usuarios presentes en la muestra.

Las características de la muestra final obtenida de la red social de Twitter son las que se muestran a continuación en la Tabla 1. Además, una visualización en forma de grafo de la muestra de la red tomada se muestra en la Figura 5.

		Entrenamiento	**Test**
Follow	Usuarios	10.029	7.086
	Enlaces	560.225	59.451
Retweet	Usuarios	9.236	2.163
	Enlaces	75.199	21.728

Tabla 1. Características de la muestra tomada de la red social de Twitter.

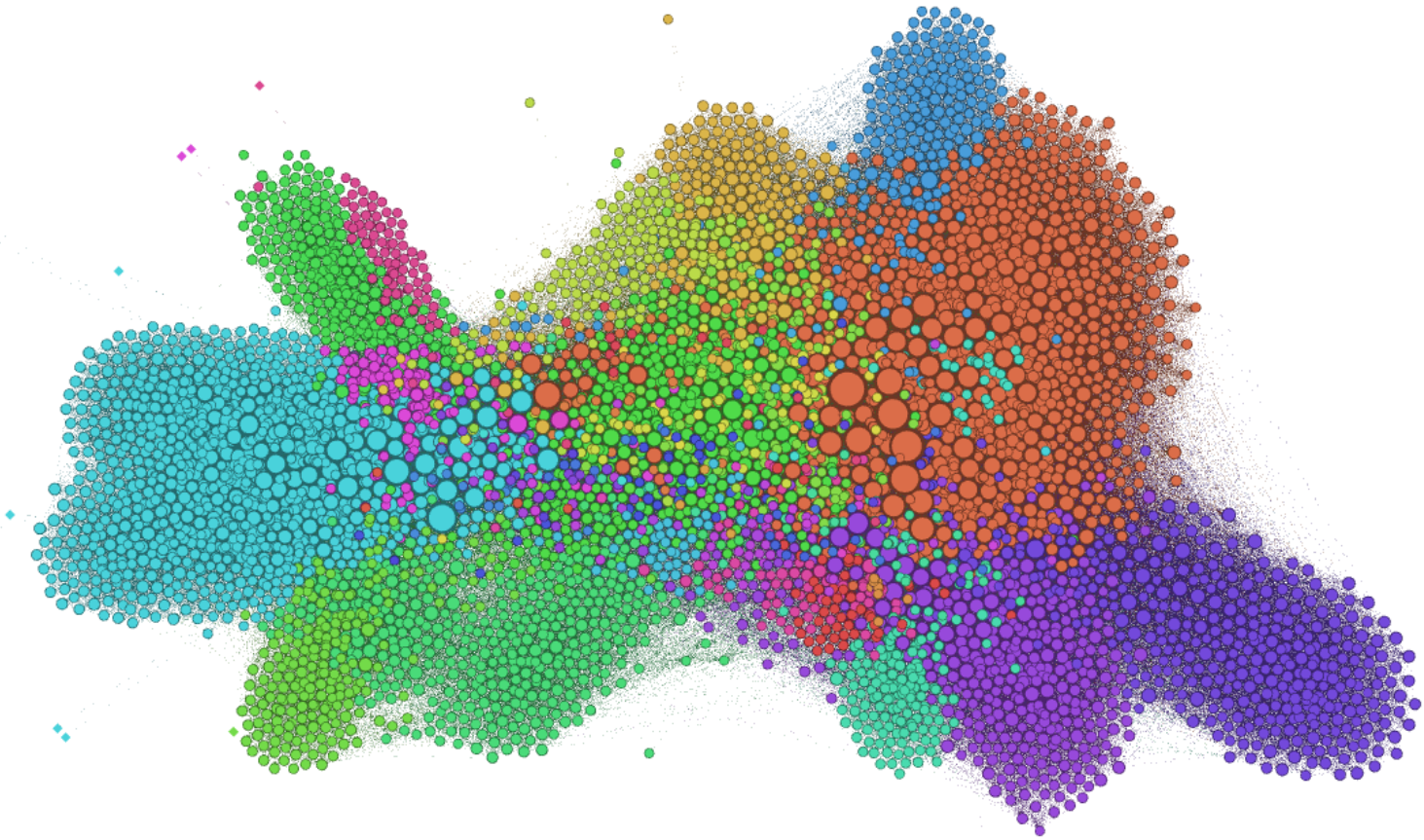

Figura 5. Representación de la muestra tomada de la red social de Twitter, donde cada color representa cada una de las comunidades identificadas con Infomap.

Toda la información relativa a la muestra tomada ha sido almacenada en una base de datos privada, de forma que los programas que requieran el uso de estos datos necesitan tener las credenciales correctas.

Entorno de trabajo

Las pruebas se han realizado en dos terminales: el ordenador local (algunas métricas) y el servidor del laboratorio de investigación (la gran mayoría de ejecuciones). El primero dispone de 32GB de RAM y un procesador AMD i5, mientras que el segundo consta de 512GB de RAM y 64 procesadores de 1.4GHz.

Espacios de atributos

Para algunas de las métricas de evaluación (principalmente las de diversidad) es necesario determinar el conjunto de subtopics o aspectos que tiene asociado cada uno de los usuarios (ERR-IA, S-Recall, etc.). En el caso de la red social de Twitter se van a utilizar dos espacios de aspectos: comunidades y hashtags.

Las comunidades de usuarios se identifican mediante algoritmos de detección que se basan en las densidades locales de enlaces. De esta forma, quedan agrupados usuarios que con mayor probabilidad pertenecen al mismo círculo, desde un grupo de amigos hasta investigadores que estudian campos relacionados. Tomando entonces las comunidades como aspectos de diversidad, lo que se plantea es medir en qué grado se están recomendando usuarios de círculos nuevos o poco conocidos para el usuario objetivo,

de forma que se le aporte más variedad y tenga la posibilidad de ampliar sus horizontes en otras direcciones.

Por el contrario, los hashtags se pueden utilizar para identificar diferentes tipos de contenido generado por los usuarios, entendiendo entonces que dos tweets con el mismo hashtag tratan del mismo tema. De esta forma, tomando los hashtags como aspectos de los usuarios se busca medir qué tan diverso es el contenido que cada usuario recomendado suele o va a generar o transmitir.

Librerías software

En este proyecto se han incorporado varias librerías para obtener la funcionalidad de algunos algoritmos, evitando tener que implementar todos los de recomendación, evaluación y detección de comunidades. El resto de algoritmos, cuya implementación no se ha tomado de ninguna librería, han sido desarrollados desde cero. Las librerías utilizadas son las siguientes:

- **RankSys**[9]: librería pública en Java implementada por el grupo de investigación de Information Retrieval Group (IRG) de la Universidad Autónoma de Madrid. Dispone de implementaciones tanto de algoritmos de recomendación como de métricas de evaluación tradicionales (acierto y error) y nuevas (novedad y diversidad). En este caso, se han tomado las implementaciones de los algoritmos de recomendación de HKV, UserkNN, ItemkNN, Popularidad y Random, y las métricas de evaluación de ERR-IA, EPC y Gini.
- **IRG projects**[10]: librería privada en Java implementada igual que la anterior por el grupo de investigación de IRG. En ella se encuentran varias implementaciones de algoritmos y métricas de evaluación, de las que se han utilizado para el trabajo la mayoría de métricas de acierto, novedad y diversidad incluidas en el estudio.

En cuanto a los algoritmos de detección de comunidades, existen numerosas librerías que implementan los diferentes métodos, pero en este caso se han utilizado las siguientes:

- **Gephi**[11]: programa en Java que contiene la implementación del algoritmo de Louvain (Blondel 2008) y un plugin para la del algoritmo Markov Cluster Algorithm (Van Dongen 2000).
- **Jung**[12]: librería Java que incorpora implementaciones como la del algoritmo de Girvan-Newman (Girvan-Newman 2004) utilizado en este estudio.
- **Criteria Comparison**[13]: proyecto Java desarrollado para la realización de experimentos del artículo de investigación (Rabbany 2013). De este proyecto, únicamente se utiliza para el presente trabajo la clase *Infomap.java*, (Rosvall 2008) que sirve de *wrapper* para ejecutar la implementación original de este algoritmo[14].
- **Jmod**[15]: un toolkit en Java que contiene la implementación del algoritmo Leading Vector de Newman (Newman 2006) así como otros basados en genética, de fuerza bruta o de movimiento de vértices.

[9] Librería RankSys (http://ranksys.org/)
[10] Information Retrieval Group (http://ir.ii.uam.es/)
[11] Herramienta Gephi (http://gephi.github.io/)
[12] Librería Jung (http://jung.sourceforge.net/)
[13] Proyecto Criteria Comparision (http://webdocs.cs.ualberta.ca/~rabbanyk/criteriaComparison/)
[14] Implementación del algoritmo de Informap (http://mapequation.org/code.html)
[15] Jmod, *toolkit* para la detección de comunidades en redes (http://tschaffter.ch/projects/jmod/)

Para el uso de estos algoritmos ha sido necesario en todos los casos realizar una adaptación de los datos de la red social a los tipos y objetos específicos de grafo de cada librería para poder ser interpretados por los algoritmos de detección de comunidades.

5.2.2 Resultados generales

La Tabla 3 recoge los resultados globales obtenidos en el estudio, mostrando únicamente el resultado de las versiones óptimas de cada algoritmo (la configuración se indica entre paréntesis).

El formato de las tablas mostradas en este capítulo es el siguiente: cada fila es un algoritmo y cada columna una métrica (agrupadas por precisión, métricas de diversidad estructural, novedad y diversidad). Los valores se muestran con un gradiente de color por métrica (i.e. columna), donde los valores con fondo verde son los mejores (el mejor se resalta en letra negrita de color blanco) y los rojos, los peores. Los algoritmos (filas) de las tablas están ordenados de forma descendiente por la métrica de precisión, y en todos ellos se establece un cutoff de 10 (sólo se tienen en cuenta los primeros 10 contactos recomendados a cada usuario) denotado como "@10". En los casos en los que al lado del valor de la métrica se muestra un asterisco (*), significa que ese valor es estadísticamente significativo frente al de la fila inferior.

La métrica de cobertura se ha aplicado en todos los casos, pero los algoritmos tienen cobertura completa sin excepción, es decir, que recomiendan a todos los usuarios de test, por lo que omitimos esta columna en las tablas de resultados. Asimismo, el análisis de las componentes fuertemente conexas en el contexto de las redes sociales es muy restrictivo, de forma que los resultados no aportan información al estudio y por eso también se omiten.

En cuanto a las dimensiones de diversidad, se han utilizado hashtags y comunidades como los aspectos propios de los usuarios (y contactos). Por un lado, en cuanto a los hashtags, se han tomado todos aquellos que cada usuario haya incluido al menos en uno de sus tweets, de forma que cada usuario puede contar con más de un aspecto y la mayoría de hashtags están presentes en varios usuarios. Por otro lado, en cuanto a las comunidades, el conjunto de entrenamiento fue sometido a los algoritmos de detección de comunidades tomando como aspectos las propias comunidades identificadas, de forma que cada usuario tendrá asociado un único aspecto, la comunidad a la que pertenece.

Se han utilizado en estas pruebas casi 17.000 hashtags como aspectos, y las comunidades detectadas han sido 6, 8 y 49 para los algoritmos de Leading Vector, Louvain e Infomap, respectivamente.

Además, existen numerosas métricas que correlacionan con otras, de forma que se ha optado por simplificar la tabla y omitir aquellas métricas que no aporten nueva información. A continuación en la Tabla 2 se muestran las métricas incluidas en la tabla de resultados, detallando aquellas con las que correlacionan (y que por tanto se omiten).

Nombre	**Métrica**	**Correlaciona con**
P	Precisión@10	
R	Recall@10	
Mod.	Modularidad de Infomap	Modularidad de grado, Modularidad de Leading Vector y de Louvain, enlaces débiles y puentes entre comunida-

Nombre	Métrica	Correlaciona con
		des obtenidas con Infomap, Louvain y Leading Vector
CCG	Coeficiente de clustering global	Coeficiente de clustering medio local
Arraigo	Arraigo	
Arraigo0	Arraigo cero	
EPC	EPC@10	
C ERR-IA	ERR-IA con las comunidades obtenidas con Infomap como subtopics	ERR-IA con las comunidades obtenidas con Leading Vector y Louvain como subtopics
HT ERR-IA	ERR-IA con los hashtags como subtopics	
C SR	Subtopic Recall @10 con las comunidades obtenidas con Infomap como subtopics	Subtopic Recall con las comunidades obtenidas con Leading Vector y Louvain
Gini	Coeficiente de Gini	

Tabla 2. Métricas mostradas en la tabla de resultados generales y métricas con las que cada una de éstas correlaciona.

En el Anexo I se muestran los resultados completos obtenidos para cada recomendador en todas las métricas seleccionadas, y en el Anexo II los resultados de la significatividad estadística para estos algoritmos para la métrica de precisión @10.

	Accuracy		Graph				Novelty	Diversity			
Recommender	P@10	R@10	Mod.	CCG	Arraigo	Arraigo0	EPC	C ERR-IA	HT ERR-IA	C SR@10	Gini
HKV (k=100, λ=150, α=40)	**0.0649***	**0.0921**	0.6677	0.1330	0.0668	1735	0.9624	**0.0643**	0.8320	**0.1282**	0.1207
User kNN (|N(u)|=50)	0.0619	0.0879	0.6790	0.1376	0.0765	1303	0.9579	0.0624	0.8462	0.1178	0.0999
BM25 (N(u)=UND, N(v)=UND, b=0.2, k=100)	0.0608	0.0919	0.6854	0.1433	0.0899	2705	0.9713	0.0637	**0.8845**	0.1194	0.2233
ExtremeBM25 (N(u)=UND, N(v)=UND, b=0.2)	0.0606	0.0916	0.6854	0.1433	0.0900	2682	0.9713	0.0636	0.8837	0.1192	0.2242
QLJM (N(u)=OUT, N(v)=IN, λ=0.5)	0.0594	0.0861	0.6799	0.1392	0.0995	173	0.9727	0.0622	0.8420	0.1165	0.1781
Item kNN (|N(u)|=700)	0.0580	0.0816	0.6816	0.1391	0.0814	1447	0.9595	0.0603	0.8140	0.1097	0.0967
Adamic (N(u)=OUT, N(v)=IN, N(w)=IN)	0.0574*	0.0825	0.6709	0.1317	0.0758	160	0.9527	0.0568	0.7616	0.1148	0.0902
FOAF (N(u)=OUT, N(v)=IN)	0.0556*	0.0785	0.6696	0.1308	0.0740	160	0.9518	0.0557	0.7375	0.1127	0.0773
Tf-idf (N(u)=OUT, N(v)=IN)	0.0495*	0.0730	0.6779	0.1481	0.1205	215	0.9924	0.0531	0.7553	0.0959	0.4395
Jaccard (N(u)=OUT, N(v)=IN)	0.0474*	0.0695	0.6811	0.1463	0.1280	217	0.9924	0.0502	0.7122	0.0916	0.3719
PurePersonalizedPageRank (r=0.8)	0.0450*	0.0679	0.6496	0.1259	0.0584	185	0.9583	0.0394	0.5208	0.1034	0.0760
PersonalizedPageRank (r=0.8)	0.0447*	0.0671	0.6494	0.1258	0.0582	185	0.9576	0.0392	0.5179	0.1024	0.0735
Closure (N(u)=OUT, N(v)=IN)	0.0417*	0.0592	0.6362	0.1430	0.0441	29003	0.9894	0.0529	0.7656	0.0846	0.1551
PageRank (r=0.6)	0.0160*	0.0250	0.5909	0.1041	0.0061	17031	0.8940	0.0111	0.2296	0.0339	0.0011
Popularity	0.0149	0.0156	0.5889	0.0992	0.0069	23808	0.8699	0.0082	0.1409	0.0274	0.0012
SALSA (authorities recommendation)	0.0149*	0.0156	0.5889	0.0992	0.0069	23808	0.8699	0.0082	0.1409	0.0274	0.0012
Rocchio	0.0103*	0.0109	0.5883	0.0981	0.0070	35677	0.9289	0.0062	0.0871	0.0188	0.0031
SALSA (hubs recommendation)	0.0028*	0.0033	**0.5856**	**0.0789**	0.0036	51779	0.9606	0.0013	0.0290	0.0059	0.0009
Random	0.0008	0.0010	0.5859	0.1078	**0.0024**	**61908**	**0.9944**	0.0005	0.0091	0.0017	**0.7906**

Tabla 3. Resultados generales obtenidos para todos los algoritmos estudiados y una selección de las métricas más interesantes.

En primer lugar, destaca cómo comparan los algoritmos más originales del presente trabajo respecto a los tomados del estado del arte. HKV no se había aplicado anteriormente a la recomendación, y es el que mejores resultados obtiene forma conjunta en todas las métricas. Además, el sistema de recomendación adaptado BM25 resulta ser muy competitivo, mejorando otros especializados en este contexto como Adamic, SALSA y PageRank. Por último, es de destacar que los algoritmos de kNN tienen asimismo buenas posiciones en la lista, de forma que se puede decir que se pueden interpretar los ítems como contactos a recomendar.

Observando los resultados de las métricas de acierto, el algoritmo de factorización de matrices HKV toma la delantera, siendo además la diferencia con el resto estadísticamente significativa y convirtiéndolo por tanto en el mejor algoritmo en cuanto al acierto. HKV es seguido por los principales algoritmos que basan sus recomendaciones en el vecindario del usuario objetivo: los algoritmos de vecinos próximos (kNN basado en usuario y en ítem), los adaptados de IR (ambas versiones de BM25 y QLJM), y Adamic. Sin embargo, no se puede saber con certeza cuáles de éstos son mejores que los otros, ya que entre ellos la diferencia no llega a ser estadísticamente significativa.

De forma más particular, se observa que de las tres versiones disponibles de PageRank, la que más acierto consigue (incluso con significatividad estadística) es la creada en el presente trabajo, aunque la mejora respecto a la versión personalizada de PageRank no es muy grande. Esto, como se ha explicado anteriormente, se ha conseguido obligando a que el camino aleatorio se realice siempre desde el usuario y volviendo al mismo sólo al llegar a un nodo del que no pueda salir, de forma que se obligue al caminante a profundizar en mayor medida en el grafo repartiendo más equilibradamente los valores de PageRank de los nodos.

Por otro lado, llama la atención que los resultados del algoritmo de popularidad sean idénticos a los de SALSA recomendando las autoridades. Al realizar el grafo bipartito propio del algoritmo, normalmente la parte izquierda (la de las autoridades) está formada por algunos de los usuarios más populares, ya que, recordando la forma de generar este grafo, se seleccionan los followees de los usuarios de la parte derecha (cercanos al usuario). De esta forma, y puesto que se trata de una pequeña muestra de la red social, los rankings generados por el algoritmo de SALSA (en la versión de recomendación de autoridades) resultan ser idénticos a los del algoritmo de popularidad.

Asimismo, se observa que Rocchio, el único algoritmo basado en contenido de los seleccionados para la comparativa, no obtiene resultados competitivos con el resto, quedando entre los últimos tres peores muy cerca de Random. En este caso, para la generación de los centroides se han utilizado los términos de los tweets de cada usuario, pero es posible que se trate de un conjunto muy grande de términos y que, con ello, no se obtengan realmente centroides representativos para los usuarios sino todos demasiado similares o bien todos muy variados y con pocas coincidencias. O es también posible que el contenido de los tweets no sea la mejor información para representar a los usuarios a la hora de elegir otros usuarios a los que recomendar seguir.

Dejando atrás las métricas de relevancia, en cuanto a las de diversidad estructural propias de los grafos se puede decir que se observa una tendencia bastante opuesta al acierto. Los algoritmos que peores resultados han obtenido en las métricas de acierto son los que maximizan las métricas de diversidad estructural. Este fenómeno se puede explicar fácilmente tomando el caso del algoritmo de recomendación aleatoria (Random). Si la recomendación se realiza de forma aleatoria, es lógico que el acierto sea muy malo, ya que se están generando los rankings sin seguir ningún criterio. Sin em-

bargo, por esta misma razón estas recomendaciones van a tender a conectar a todos los usuarios con todos, favoreciendo en gran medida algunas métricas como la modularidad, por conectar usuarios de diferentes comunidades, y el arraigo, por unir a muchos usuarios lejanos en la red. En otras palabras, es fácil proporcionar diversidad renunciando al acierto.

Según esta misma tendencia, los algoritmos basados en la recomendación por vecindario que tan buenos resultados de acierto han tenido, son los que menos favorecen las métricas de grafo. Sin embargo, destaca Closure, que a pesar de basarse en vecindarios consigue buenos resultados de modularidad y arraigo, aunque el coeficiente de clustering es de los peores; y HKV, que es de los únicos que obtiene buenos resultados de acierto y consigue mantenerse en unos valores normales en cuanto a las métricas de diversidad estructural.

Como se puede observar, conseguir un algoritmo que maximice el acierto y además mejore ciertas características de la red social es muy difícil. Normalmente, las recomendaciones que el usuario considera como relevantes son la de aquellos contactos que están no muy lejos de su vecindario. Sin embargo, para favorecer las métricas de diversidad estructural es necesario romper con los vecindarios y comunidades para que todos estén cerca de todos y se obtengan mejoras como una mayor velocidad de propagación de la información.

En definitiva, el acierto y la diversidad son dimensiones que tienden a presentar en general una relación de contrapartida entre sí. Una aproximación para conseguir la armonía entre los dos objetivos es la utilización de algoritmos de reordenación multiobjetivo que, partiendo de las recomendaciones generadas inicialmente por un algoritmo orientado a maximizar el acierto, reordenen las recomendaciones para buscar la mejora de cierta característica de la red social. En la sección 5.1.5 explicaremos y mostraremos los efectos de algunos algoritmos de reordenación en esta línea.

Finalmente, en cuanto a las métricas de novedad y diversidad, se observan dos tendencias principales: EPC y Gini frente a las versiones de ERR-IA y Subtopic Recall.

Por un lado, la métrica de EPC de novedad calcula la probabilidad a priori de que un usuario no haya interactuado con los contactos recomendados, mientras que la métrica Gini de diversidad valora en qué medida se recomiendan todos los usuarios como contacto a algún usuario u otro de forma equilibrada. Aquí destaca (trivialmente) la recomendación aleatoria en ambas métricas, de la misma forma que lo hizo en el caso de las de grafo, ya que, como se ha explicado anteriormente, con este algoritmo todos los contactos tienden a tener la misma probabilidad de ser recomendados (no exactamente igual porque no se recomiendan contactos que el usuario objetivo ya siga, por lo que la probabilidad tiene un pequeño sesgo inverso a la popularidad). Por el contrario, destacan negativamente los algoritmos similares a popularidad (Popularidad, SALSA y PageRank), que es normal que ocurra puesto que EPC es una métrica que busca precisamente la anti-popularidad.

Por otro lado, las métricas de ERR-IA y Subtopic Recall tienden a ir muy relacionadas con la precisión, ya que la primera utiliza grados de relevancia y la segunda sólo tiene en cuenta los contactos que han sido relevantes para el usuario. Centrando el estudio en el tipo de subtopics utilizados en cada caso, se observa que con comunidades el mejor resultado es el de HKV, tanto en ERR-IA como en Subtopic Recall, pero usando los hashtags los algoritmos que lo maximizan son las versiones de BM25. Esto indica que mientras HKV recomienda contactos de otras comunidades diferentes a las del usuario objetivo, los algoritmos de BM25 consiguen que la diversidad esté en el conte-

nido (tweets) que generan los contactos recomendados (entendiendo que los hashtags son representativos de los tweets en los que se utilizan).

En resumen, dependiendo del tipo de evaluación que se pretenda optimizar (acierto, novedad, diversidad, métricas de diversidad estructural), puede ser mejor o peor utilizar ciertos algoritmos de recomendación. Sin embargo, de forma general se puede decir que los algoritmos de HKV, kNN, QLJM y los de BM25 consiguen resultados muy buenos en cuanto a acierto y diversidad, aunque se queden atrás en las métricas de diversidad estructural.

5.2.3 Optimización de los nuevos algoritmos

Al realizar los barridos de los parámetros a establecer para cada uno de los algoritmos de recomendación implementados, hemos identificado ciertas tendencias en los resultados que podían optimizarse para crear nuevas versiones de los algoritmos en cuestión.

De esta forma, en este apartado se justifican los dos casos más destacados de la creación de las nuevas variaciones: BM25 y PageRank personalizado.

BM25 → ExtremeBM25

Al realizar el barrido sobre el parámetro lambda del algoritmo BM25, se ha observado una tendencia en los resultados que podía ser maximizada: los mejores resultados se obtienen cuando el parámetro *k* tiende a infinito. Esta tendencia se puede observar a continuación en la Tabla 4.

		b						
		0.01	0.1	0.2	0.4	0.6	0.8	1.0
k	0.5	0.0507	0.0527	0.0539	0.0561	0.0566	0.0562	0.0551
	1	0.0520	0.0545	0.0563	0.0580	0.0568	0.0552	0.0523
	1.5	0.0527	0.0553	0.0575	0.0588	0.0565	0.0538	0.0496
	2	0.0529	0.0559	0.0582	0.0589	0.0562	0.0525	0.0473
	2.5	0.0532	0.0563	0.0589	0.0590	0.0556	0.0516	0.0449
	3	0.0531	0.0568	0.0591	0.0591	0.0555	0.0506	0.0431
	10		0.0584	0.0605				
	100		0.0589	**0.0608**				
	1000		0.0589	0.0606				
	10000		0.0589	0.0606				
	∞		0.0589	0.0606				

Tabla 4. Barrido realizado sobre los dos parámetros del algoritmo BM25.

En consecuencia a los resultados obtenidos, se ha calculado el límite cuando *k* tiende a infinito de la fórmula de BM25 y se ha obtenido una variación nueva de este algoritmo, ExtremeBM25, que depende de un único parámetro, *b*, y que maximiza la calidad de BM25, obteniendo el resultado óptimo mostrado anteriormente en la tabla general de resultados (Tabla 3).

PersonalizedPageRank → PurePersonalizedPageRank

En este caso, el planteamiento de una nueva versión del algoritmo de PageRank personalizado surgió a raíz del análisis de los principios que sigue este algoritmo para realizar las recomendaciones.

En el contexto de las redes sociales, existe un sesgo muy destacable causado por la diferencia de entre el tamaño del vecindario de los usuarios. De forma intuitiva, un usuario con muchos enlaces entrantes hace que los nodos más cercanos concentren la gran cantidad de PageRank, de forma que la mayoría de usuarios que se recomiendan tienen poco PageRank (ya que normalmente existe un enlace con los usuarios cercanos al objetivo), y se obtiene un ranking donde los contactos recomendados tienen poca diferencia entre ellos. Por el contrario, el caso sería muy diferente si se trata de un usuario que apenas tiene enlaces de entrada, ya que se reparte de forma más escalonada el PageRank entre los contactos cercanos y no tan cercanos, obteniendo un abanico mayor de usuarios para recomendar.

Omitiendo entonces los enlaces de entrada al usuario objetivo y añadiendo enlaces de los sumideros a éste, se consigue mejorar ligeramente la versión original de PageRank personalizado. En la tabla que se muestra a continuación (Tabla 5) se muestra la comparación entre las tres versiones de PageRank, la básica, la personalizada y la nueva creada. Asimismo, en la Figura 6 se representa esta comparación para observar los resultados de forma más visual, donde se observa que aunque es poca, se mejora la precisión del algoritmo de PageRank personalizado con el nuevo planteamiento.

		PR	PPR	PPPR
r	0.1	0.0120	0.0171	0.0195
	0.2	0.0137	0.0287	0.0298
	0.3	0.0155	0.0360	0.0366
	0.4	0.0147	0.0409	0.0412
	0.5	0.0152	0.0435	0.0438
	0.6	0.0160	0.0448	0.0450
	0.7	0.0158	0.0454	0.0453
	0.8	0.0159	0.0458	**0.0460**
	0.9	0.0157	0.0456	0.0456
	0.95		0.0458	0.0456
	0.999		0.0456	0.0456

Tabla 5. Barrido realizado de P@10 sobre el parámetro de los algoritmos de PageRank (PR), PersonalizedPageRank (PPR) y PurePersonalizedPageRank (PPPR).

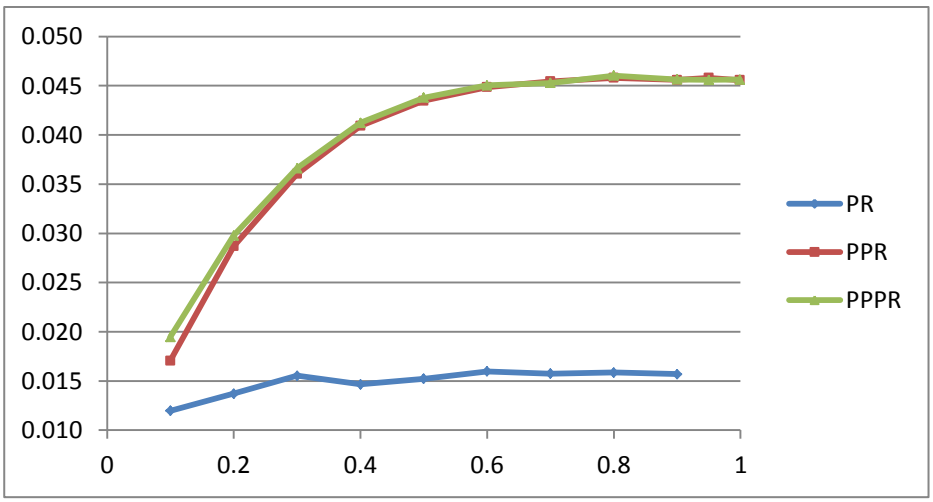

Figura 6. Representación de los barridos de P@10 realizados para los algoritmos de PageRank (PR), PersonalizedPageRank (PPR) y PurePersonalizedPageRank (PPPR).

5.2.4 Direccionalidad de los arcos

Una de las motivaciones del presente trabajo es la de responder a algunas preguntas que surgen relacionadas con la recomendación en redes sociales, concretamente en el contexto de la red social de Twitter.

Debido a que Twitter es una red social dirigida y asimétrica, cabe considerar cuatro casos en la relación entre dos usuarios: uno sigue al otro, el otro sigue al primero, ambas cosas, o ninguna (Figura 7).

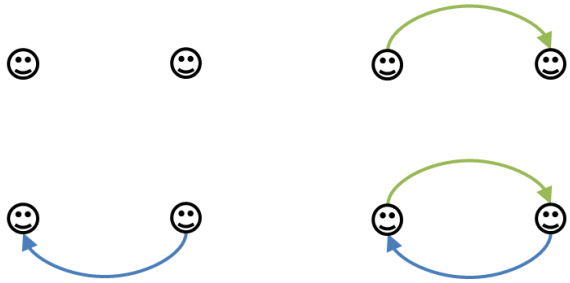

Figura 7. Posibles relaciones entre usuarios.

A la hora de realizar recomendaciones, muchos de los sistemas de recomendación toman como principal fuente de información la topología de la red, es decir, los enlaces entrantes y/o salientes que tiene el usuario objetivo. Así, éstos interpretan que nuestros gustos y preferencias están ligados a los usuarios con los que estamos conectados, es decir, a nuestro vecindario.

Existen diferentes combinaciones dependiendo de la dirección del enlace que se toma para seleccionar el vecindario de los usuarios, tanto candidatos como el objetivo. De forma general, se pueden considerar tres tipos de vecindarios según el tipo de enlace que une a los usuarios: IN, OUT y UND (*undirected*) (Figura 8).

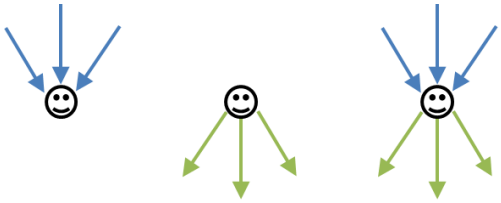

Figura 8. Tipos de vecindarios.

Entonces, ***¿qué define mejor nuestras preferencias, nuestros* followers *(las personas que nos siguen), nuestros* followees *(las personas a las que seguimos), o la unión de ambos?***

Para responder a esta pregunta, comparamos el comportamiento de los algoritmos de recomendación que se basan en la topología de la red (es decir, que utilizan vecindarios) con las diferentes combinaciones posibles, observando cuál es más efectiva en términos de precisión. En la Tabla 6 se muestran los resultados obtenidos (el encabezado oscuro es para el tipo de vecindario del usuario objetivo y el claro, para el de los candidatos; el gradiente de color es relativo a cada recomendador, es decir se aplica por filas de la tabla, destacando con blanco su valor máximo).

	OUT			UND			IN		
Recommender	**OUT**	**UND**	**IN**	**OUT**	**UND**	**IN**	**OUT**	**UND**	**IN**
Adamic (w IN)	0.0337	0.0429	**0.0586**	0.0328	0.0510	0.0572	0.0345	0.0532	0.0531
Closure	0.0296	0.0316	**0.0431**	0.0293	0.0369	0.0429	0.0299	0.0390	0.0428
ExtremeBM25 (0.2)	0.0195	0.0602	0.0559	0.0200	**0.0606**	0.0560	0.0247	0.0556	0.0518
Jaccard	0.0362	0.0285	0.0482	0.0370	0.0420	**0.0551**	0.0336	0.0428	0.0472
MCN	0.0306	0.0384	**0.0567**	0.0301	0.0473	0.0566	0.0348	0.0537	0.0542
QLJM (0.9)	0.0367	0.0414	0.0610	0.0365	0.0531	**0.0625**	0.0378	0.0573	0.0579
TFIDF	0.0425	0.0453	0.0516	0.0434	0.0541	**0.0565**	0.0381	0.0510	0.0504

Tabla 6. Resultados obtenidos de P@10 para los diferentes tipos de vecindarios.

Los resultados se pueden analizar desde dos puntos de vista: uno global, donde se observan las combinaciones que más destacan en general, y uno horizontal, donde se identifican los tipos de vecindario que más aportan precisión a cada algoritmo de recomendación particular.

Por un lado, las combinaciones que más destacan de forma general son dos: OUT-IN y UND-IN, pero no se puede dar un ganador claro entre estas dos opciones.

La primera de estas combinaciones, OUT-IN, es la que se esperaría que funcionase mejor, ya que se busca cerrar relaciones transitivas (*u* sigue a *w*, *w* sigue a *v*, entonces a *u* le puede interesar seguir a *v*). De forma más analítica, se selecciona como vecindario del usuario objetivo sus followees, y para los candidatos, sus followers, por lo que la información que define al usuario al que se le va a realizar la recomendación depende principalmente de aquellos usuarios a los que ha decidido seguir, mientras que en el caso de los candidatos, éstos son definidos por la gente que los sigue.

En cuanto a la opción UND-IN, es parecida a OUT-IN pero tomando para el vecindario del usuario objetivo sus followers además de sus followees, de forma que en OUT-IN se utiliza un subconjunto del vecindario que usa éste. Así, la información que define al usuario objetivo con esta combinación depende de todos aquellos otros usuarios con los que esté conectado (se usa toda la información del usuario objetivo).

Por otro lado, prestando atención a los resultados de cada algoritmo, se puede observar que los resultados son notablemente mejores cuando se toma como vecindario de los usuarios candidatos los enlaces de entrada (IN), y se obtienen resultados muy bajos si se toman los de salida (OUT). Con esto se puede decir que de forma general, al usuario candidato lo definen mejor sus followers. Sin embargo, no se puede determinar con exactitud el tipo de vecindario que mejor define al usuario objetivo.

5.2.5 Optimización directa de métricas

Todos los algoritmos de recomendación que se han utilizado en este estudio realizan sus recomendaciones de forma individual para cada usuario, buscando maximizar el acierto de las recomendaciones. Sin embargo, como se ha explicado anteriormente, en este trabajo se plantea una nueva visión en la que se busque mejorar cierta propiedad de la red global, como el coeficiente de clustering o la modularidad. De esta forma, se realiza un equilibrio multiobjetivo con dos objetivos contrapuestos: recomendar a los usuarios aquellos contactos que más probabilidad tienen de interesarles, y mejorar alguna característica concreta del grafo, como establecer enlaces nuevos entre comunidades para que se mejore la velocidad de propagación de la información en la red.

Este equilibrio se consigue mediante el parámetro λ, que determina el peso que se le da al ranking inicial frente al reordenado, de forma que si vale 0 no se realiza ninguna reordenación, y si vale 1 se realiza una reordenación completa que ignora el orden original definido por el algoritmo de recomendación inicial (aunque únicamente se reordenaría el top definido de contactos a tener en cuenta).

Para mostrar el efecto de los algoritmos de reordenación, seleccionamos HKV por ser el algoritmo que mejor acierto ha mostrado en nuestros experimentos comparativos anteriores, y creamos nuevas recomendaciones mediante la reordenación del top 50 del ranking de contactos inicial de este algoritmo. Como métricas objetivo para la reordenación, hemos seleccionado la modularidad (utilizando las comunidades detectadas con Infomap) y la diversidad mediante Gini.

Los resultados obtenidos del experimento se muestran en las Tablas 7 y 8.

		Accuracy			Graph				Novelty	Diversity			
Recommender		P@10	R@10	nDCG@10	Mod.	CCG	Arraigo	Arraigo0	EPC	C ERR-IA	HT ERR-IA	SR@10	Gini
original		**0.0649**	**0.0921**	**0.0914**	**0.6677**	**0.2654**	**0.0668**	**1735**	**0.9624**	**0.0643**	**0.8320**	**0.1282**	**0.1207**
Infomap	0	0.0649	0.0921	0.0914	0.6677	0.2654	0.0668	1735	0.9624	0.0643	0.8320	0.1282	0.1207
	0.1	0.0645	0.0917	0.0904	0.6649	0.2650	0.0660	1771	0.9620	0.0617	0.8253	0.1288	0.1187
	0.2	0.0639	0.0904	0.0886	0.6593	0.2641	0.0645	1819	0.9613	0.0580	0.7972	0.1287	0.1151
	0.3	0.0631	0.0892	0.0871	0.6536	0.2632	0.0630	1885	0.9607	0.0547	0.7777	0.1278	0.1121
	0.4	0.0616	0.0872	0.0849	0.6457	0.2617	0.0608	2005	0.9600	0.0512	0.7592	0.1261	0.1083
	0.5	0.0599	0.0851	0.0823	0.6368	0.2600	0.0581	2173	0.9596	0.0470	0.7285	0.1230	0.1046
	0.6	0.0566	0.0804	0.0778	0.6263	0.2578	0.0547	2424	0.9594	0.0416	0.6885	0.1179	0.1000
	0.7	0.0546	0.0767	0.0750	**0.6210**	0.2565	0.0530	2602	0.9594	0.0383	0.6681	0.1141	0.0983
	0.8	0.0546	0.0767	0.0750	**0.6210**	0.2565	0.0530	2602	0.9594	0.0383	0.6681	0.1141	0.0983
	0.9	0.0546	0.0767	0.0750	**0.6210**	0.2565	0.0530	2602	0.9594	0.0383	0.6681	0.1141	0.0983
	1	0.0381	0.0538	0.0498	0.6211	**0.2501**	**0.0484**	**3548**	0.9653	0.0214	0.4313	0.0830	0.0820
Infomap-Gini	0	0.0649	0.0921	0.0891	0.6677	0.2654	0.0668	1735	0.9624	0.0569	0.7943	0.1282	0.1207
	0.1	0.0642	0.0912	0.0878	0.6608	0.2641	0.0665	1820	0.9633	0.0540	0.7827	0.1286	0.1252
	0.2	0.0635	0.0906	0.0868	0.6567	0.2632	0.0662	1895	0.9639	0.0522	0.7732	0.1286	0.1275
	0.3	0.0626	0.0893	0.0858	0.6529	0.2625	0.0659	1980	0.9644	0.0508	0.7658	0.1275	0.1297
	0.4	0.0622	0.0887	0.0850	0.6496	0.2618	0.0655	2062	0.9649	0.0494	0.7574	0.1267	0.1316
	0.5	0.0614	0.0876	0.0840	0.6466	0.2612	0.0651	2144	0.9653	0.0483	0.7487	0.1260	0.1334
	0.6	0.0606	0.0871	0.0833	0.6438	0.2606	0.0648	2222	0.9657	0.0471	0.7426	0.1250	0.1349
	0.7	0.0600	0.0858	0.0823	0.6414	0.2600	0.0644	2293	0.9661	0.0460	0.7355	0.1241	0.1363
	0.8	0.0594	0.0848	0.0813	0.6392	0.2595	0.0641	2376	0.9664	0.0450	0.7293	0.1230	0.1375
	0.9	0.0589	0.0842	0.0807	0.6372	0.2590	0.0637	2451	0.9667	0.0442	0.7235	0.1223	0.1386
	1	0.0583	0.0833	0.0800	0.6353	0.2585	0.0634	2533	0.9671	0.0435	0.7179	0.1209	**0.1397**

Tabla 7. Resultados obtenidos para los algoritmos de reranking según una selección de las métricas más interesantes (Parte 1 de 2).

		Accuracy			Graph				Novelty	Diversity			
Recommender		P@10	R@10	nDCG@10	Mod.	CCG	Arraigo	Arraigo0	EPC	C ERR-IA	HT ERR-IA	SR@10	Gini
original		**0.0649**	**0.0921**	**0.0914**	**0.6677**	**0.2654**	**0.0668**	**1735**	**0.9624**	**0.0643**	**0.8320**	**0.1282**	**0.1207**
Comm-Gini	0	0.0649	0.0921	0.0914	0.6677	0.2654	0.0668	1735	0.9624	0.0643	0.8320	0.1282	0.1207
	0.1	0.0651	0.0920	0.0914	0.6679	0.2655	0.0669	1725	0.9626	0.0642	**0.8326**	0.1283	0.1212
	0.2	0.0650	0.0916	0.0909	0.6685	0.2656	0.0672	1730	0.9630	0.0637	0.8251	0.1283	0.1224
	0.3	0.0653	0.0920	0.0912	0.6690	0.2657	0.0676	1733	0.9634	0.0633	0.8273	0.1286	0.1236
	0.4	0.0652	0.0917	0.0907	0.6698	0.2659	0.0682	1744	0.9641	0.0624	0.8235	0.1287	0.1253
	0.5	0.0651	0.0914	0.0904	0.6709	0.2661	0.0689	1758	0.9649	0.0614	0.8237	0.1279	0.1276
	0.6	0.0650	0.0920	0.0901	0.6723	0.2664	0.0698	1786	0.9662	0.0602	0.8204	0.1277	0.1301
	0.7	0.0637	0.0907	0.0887	0.6730	0.2665	0.0706	1910	0.9677	0.0581	0.8130	0.1256	0.1323
	0.8	0.0630	0.0896	0.0872	0.6716	0.2663	0.0710	2116	0.9694	0.0552	0.8007	0.1234	0.1328
	0.9	0.0615	0.0881	0.0850	0.6688	0.2656	0.0712	2397	0.9713	0.0513	0.7801	0.1208	0.1329
	1	0.0599	0.0849	0.0809	0.6649	0.2647	0.0708	2727	**0.9724**	0.0465	0.7410	0.1177	0.1336
XQuAD	0	0.0649	0.0921	0.0914	0.6677	0.2654	0.0668	1735	0.9624	0.0643	0.8320	0.1282	0.1207
	0.1	0.0651	0.0931	0.0917	0.6676	0.2654	0.0668	1731	0.9624	**0.0645**	0.8322	0.1300	0.1211
	0.2	**0.0656**	0.0935	**0.0919**	0.6675	0.2653	0.0668	1711	0.9625	0.0640	0.8297	0.1310	0.1216
	0.3	0.0655	**0.0937**	0.0917	0.6668	0.2650	0.0666	1705	0.9626	0.0629	0.8226	0.1317	0.1224
	0.4	0.0649	0.0936	0.0905	0.6649	0.2643	0.0661	1767	0.9627	0.0602	0.8051	0.1330	0.1234
	0.5	0.0645	0.0930	0.0892	0.6641	0.2637	0.0657	1837	0.9631	0.0575	0.7882	0.1337	0.1247
	0.6	0.0644	0.0930	0.0890	0.6640	0.2634	0.0657	1884	0.9635	0.0563	0.7847	**0.1343**	0.1268
	0.7	0.0640	0.0920	0.0882	0.6639	0.2633	0.0657	1920	0.9638	0.0553	0.7786	0.1337	0.1281
	0.8	0.0638	0.0921	0.0880	0.6638	0.2632	0.0656	1957	0.9640	0.0547	0.7749	0.1333	0.1288
	0.9	0.0638	0.0923	0.0877	0.6636	0.2631	0.0656	1983	0.9641	0.0541	0.7699	0.1333	0.1294
	1	0.0637	0.0925	0.0876	0.6635	0.2630	0.0656	1997	0.9642	0.0538	0.7658	0.1332	0.1299

Tabla 8. Resultados obtenidos para los algoritmos de reranking según una selección de las métricas más interesantes (Parte 2 de 2).

De forma general se observa que efectivamente cada reranker maximiza aquellas métricas en las que se enfoca mejorar: Infomap destaca en modularidad, Infomap-Gini en modularidad y Gini, Comm-Gini en novedad y XQuAD en diversidad. A continuación se realiza un análisis de los resultados estudiando cada conjunto de métricas de forma individual.

Es común que al realizar un reordenamiento de los rankings de recomendaciones se tienda a empeorar el acierto, ya que se mueven elementos de posiciones bajas a posiciones altas y viceversa, rompiendo la ordenación que el algoritmo de recomendación había estimado cuidadosamente. Sin embargo, en este caso XQuAD consigue mejorar, aunque muy ligeramente, el acierto del recomendador base cuando la ponderación está en 0.2-0.3.

Todos los rerankers (excepto Comm-Gini) tienden a mejorar los resultados del algoritmo base en las métricas de grafo, destacando el reranker de Infomap que consigue maximizar los valores. Esto se debe a la naturaleza de este reranker, que es el único que se centra completamente en buscar la mejora de la modularidad afectando positivamente a los resultados para el resto de las métricas de grafo. Cabe destacar además que el algoritmo de reranking Infomap-Gini obtiene buenos resultados en modularidad, pero sin embargo no llega a los valores de Infomap. Esto se debe a que este reranker busca maximizar la modularidad igual que Infomap, pero está limitado por Gini, de forma que realiza el reordenamiento de los rankings buscando mejorar la modularidad pero sólo si los cambios realizados mejoran y no empeoran la diversidad de Gini.

En cuanto a la novedad, destaca sobre el resto la mejora que realiza Comm-Gini, aunque el resto de rerankers tienden a mantener o mejorar ligeramente el valor del *baseline* excepto Infomap, que lo empeora un poco.

Por último, observando los resultados de las métricas de diversidad se observa que todos los rerankers tienden a empeorar en general el *baseline* excepto XQuAD, que maximiza la métrica de Subtopic Recall, e Infomap-Gini y Comm-Gini que mejoran en gran medida el valor de Gini (y el primero maximiza su valor).

A partir de estos resultados, puede resultar interesante ver si la diversidad estructural corresponde con diversidad en la propagación de información. Para ello, se en el siguiente apartado se compara una versión optimizada para diversidad estructural contra otra no optimizada.

5.2.6 Diversidad en la propagación de información

A lo largo de este trabajo hemos propuesto y observado diferentes métricas de novedad y diversidad de la recomendación, y en particular hemos explorado la definición de una perspectiva de diversidad estructural basada en la noción de enlaces débiles, que a su vez se basa en la detección de comunidades basada en modularidad. Ante nuestra propuesta se plantea de forma natural la pregunta: ¿es útil para los usuarios que les sean recomendados enlaces débiles? El interés de recomendar enlaces débiles encuentra motivación en la literatura relativa a esta noción (Granovetter 1973), en la que se ha analizado o especulado sobre el beneficio de los enlaces débiles en una red social: promueven la velocidad a la que se propaga la información, acortan la distancia promedio entre usuarios de la red, y aportan en general un valor estratégico desde diferentes puntos de vista.

A fin de abordar no obstante explícitamente la pregunta sobre la utilidad de los enlaces débiles, nos planteamos el posible efecto que la diversidad estructural, entendida como la presencia de enlaces débiles (es decir, la diversidad de comunidades entre los contactos de los usuarios), puede tener la diversidad de la información que fluye a través de estos contactos. Fomentar el equilibrio entre la personalización y la diversidad de la información a la que son expuestos los usuarios en las redes sociales apuntaría a paliar el denominado efecto "filter bubble" muy debatido en años recientes (citar libro de Eli Pariser), en el que un exceso de personalización puede hacer que los usuarios sólo "escuchen lo que les gusta oír" con el consiguiente empobrecimiento de su visión de la realidad.

Para nuestras pruebas nos apoyamos en los tweets disponibles en la descarga de datos de Twitter con la que hemos realizado los experimentos que hemos documentado en las secciones anteriores. Concretamente, para medir la diversidad en la propagación de información en una red, simulamos la publicación y retweeteo de tweets a través de la muestra de Twitter aumentada con los contactos recomendados, es decir, como si los usuarios aceptasen todas las top *k* recomendaciones. Para esto, utilizamos los tweets presentes en la muestra y establecemos un mecanismo de publicación/retweet simulado cuyo funcionamiento se ha explicado en el apartado 4.6. Dado un flujo simulado de tweets a través de la red, podemos medir en cada momento del proceso la diversidad de la información que transmiten los tweets que han llegado a cada usuario. Esta diversidad se puede medir, por ejemplo, en términos de los hashtags que aparecen en los tweets.

Existen variaciones donde se seleccionan los tweets más recientes que tengan al menos uno de los 500 hashtags más frecuentes en la colección (variación $top500$), y otros donde además de esto se eliminan los tweets seleccionados que no tengan un hashtag de los 200 más utilizados en el subconjunto de la simulación (variación $posttop200$).

Para que los efectos de diferentes algoritmos de recomendación sean perceptibles y las simulaciones terminen en un tiempo razonable, seleccionamos un subconjunto de los enlaces de entrenamiento. Así, seleccionamos aleatoriamente la mitad de los usuarios (manteniendo la misma comunidad a la que pertenecían en la muestra completa), y 5 de sus propios tweets de la muestra para cada usuario (que contengan al menos un hashtag), que serán los que propagarán por la red (parámetro $nTweets$). La muestra resultante consta de 5.030 usuarios y 7.086 enlaces follow entre ellos.

Realizamos simulaciones para dos conjuntos de algoritmos: una selección de sistemas de recomendación con los que se ha trabajado en las pruebas offline, y algoritmos de reranking con su baseline. Para el primer caso se han realizado numerosas pruebas, pero ninguna de ellas ha mostrado resultados relevantes (apenas se veía diferencia entre las curvas). Esto puede ser debido a que la mayoría de información se propague por los enlaces de entrenamiento (comunes para todos) y no por los nuevos incluidos por cada recomendador.

En cuanto a las simulaciones realizadas para los rerankers, se ha tomado como baseline el algoritmo de HKV y las versiones de $\lambda = 0.5$ y $\lambda = 0.9$ de todos ellos analizados en el apartado anterior (Infomap, Comm-Gini, Infomap-Gini y XQuAD). El objetivo en este caso es observar cómo se propaga la información y ver, desde el punto de vista de la diversidad, si los rerankers mejoran al baseline y en qué medida.

Respecto a la configuración de las simulaciones, el resultado más relevante se ha obtenido simulando que cada usuario lee cada iteración un único tweet y tiene una probabilidad de 0.5 de hacer retweet del mismo ($r = 0.0$ (mínimo un tweet), $p = 0.5$). Además, se ha utilizado la versión en la que se seleccionan los tweets con algún hashtag de los 500 más utilizados en la muestra.

En la gráfica Figura 9 se muestran los resultados. Se trata de una visión temporal de la progresión del flujo de la información a medida que avanza la simulación. En cada punto de la simulación se mide la diversidad promedio de la información que ha recibido cada usuario, en términos de hashtags según la métrica Gini de diversidad. Con esto se puede comparar lo rápido o lento que avanza la diversidad en diferentes perspectivas.

Como se puede observar, todos los algoritmos de reranking tienen la curva de $\lambda = 0.9$ por encima de la de $\lambda = 0.5$ y éstas (en general) por encima del baseline, de forma que todos tienden a mejorar la diversidad de hashtags respecto al baseline. Se ve por tanto que la diversidad estructural mayor o menor se corresponde consistentemente con la diversidad en la información que se propaga, medible en términos de hashtags. A mayor diversidad estructural (dada por una reordenación más agresiva), mayor diversidad en la información propagada.

XQuAD obtiene los peores resultados, consiguiendo en el caso de $\lambda = 0.5$ incluso empeorar ligeramente el baseline. Sin embargo, en términos de potenciar al máximo la diversidad de información, cabe destacar que el nuevo algoritmo de reranking con objetivo dual (Infomap-Gini con $\lambda = 0.9$) consigue los mejores resultados. No sólo mejora en gran medida al baseline, sino que además supera también a los algoritmos de reranking de Infomap y Comm-Gini. Esto sugiere que se pueden obtener mejores resultados de diversidad de información si se optimizan de forma conjunta Gini e Infomap en lugar de optimizarlas de forma individual.

Manteniendo los mismos parámetros de la simulación para medir otra métrica, Comm-Tweets (proporción de comunidades de las que los usuarios en media reciben tweets), se realiza únicamente las 100 primeras iteraciones y se hace la media de 50 ejecuciones, de forma que se pueda tener una visión en más detalle de las primeras fases de la simulación. De los resultados obtenidos, destaca la gráfica que representa la proporción de comunidades a las que pertenecen los tweets que reciben los usuarios. El resultado obtenido se muestra en la Figura 10.

En las primeras iteraciones se observa que, como es de esperar, el reranker de Infomap toma la delantera, seguido inmediatamente por el algoritmo de Infomap-Gini. Inicialmente ambos están bastante alejados del resto de rerankers, pero a medida que aumentan las iteraciones, todos los algoritmos (incluyendo el baseline) tienden a agruparse reduciendo completamente las posibles diferencias entre ellos. Esto indica que los algoritmos que buscan maximizar la modularidad en las primeras iteraciones consiguen su objetivo, que es conectar diferentes comunidades entre sí. Sin embargo, el resto de algoritmos no tarda en ponerse a la par, lo que indica que la distancia entre comunidades no es muy grande.

Es interesante señalar además que ya a partir de las 100 iteraciones se empieza a observar que todos los algoritmos tienden a llegar hasta algo más que el 80% de las comunidades. Esto puede ser debido a que, al realizar la reducción de usuarios, es posible que un 20% de las comunidades que quizás antes eran pequeñas se hayan quedado con muy pocos representantes, y que además estos no estén conectados con el resto de comunidades. Por ello, aunque la propagación de información llegue a su final, nunca se alcanzará el 100% de información recibida de comunidades por los usuarios.

Se probaron otras versiones para las simulaciones, como forzar a introducir en la red todos los tweets de los usuarios antes de que pudiesen retwittear, de forma que se evitase que los usuarios que tenían muchos enlaces de entrada tardasen mucho en propagar sus tweets propios; o incluir caducidad en los tweets a retweetear, de forma que solo los recibidos en la iteración anterior pudiesen ser retweeteados. En ambos casos, no se obtuvieron resultados relevantes.

Con los resultados obtenidos en estas simulaciones, mostramos indicios empíricos (con las limitaciones conocidas) de que la diversidad estructural efectivamente fomenta la diversidad de información propagada, justificando entonces la importancia de las métricas de diversidad estructural así como la utilización de rerankers que la fomentan.

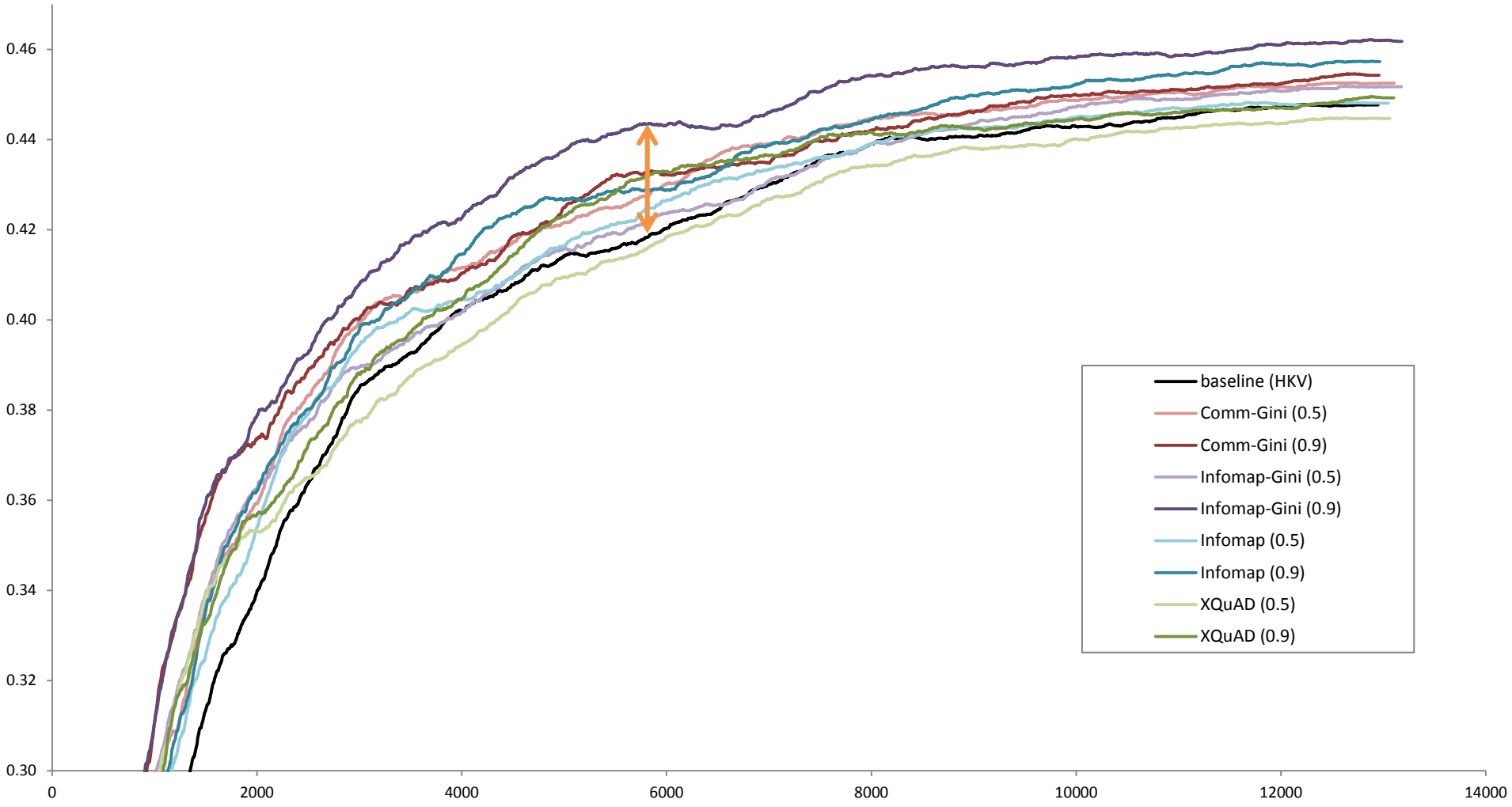

Figura 9. Resultados de HT-Gini para las simulaciones de los rerankers y el baseline, con $r = 0.0$ y $p = 0.5$. La flecha naranja señala la diferencia entre el baseline (línea color negro) respecto al reranker Infomap-Gini (0.9) (línea de color morado oscuro).

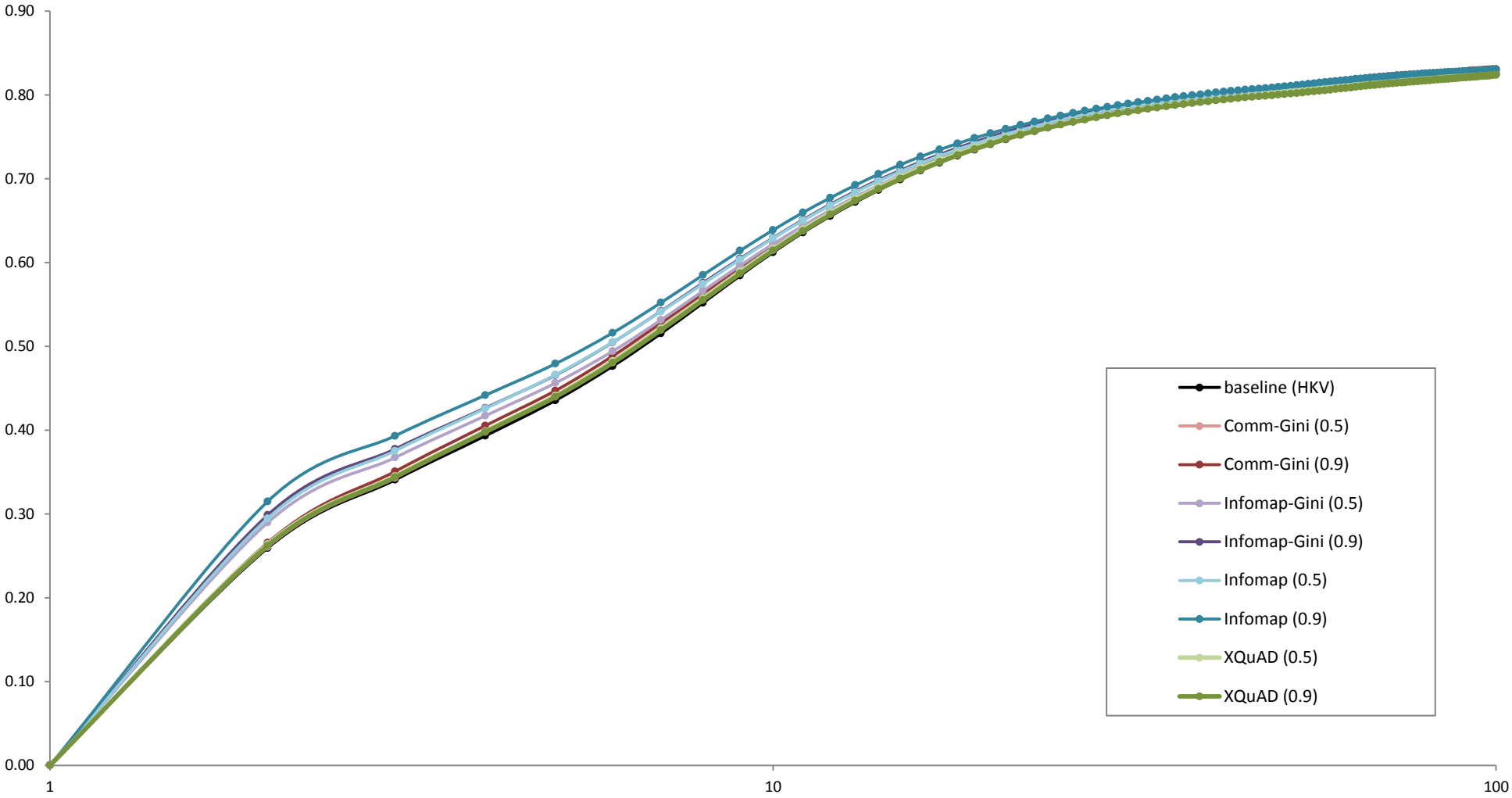

Figura 10. Resultados de Comm-Tweets para las simulaciones de los rerankers y el baseline, con $r = 0.0$, $p = 0.5$, 100 iteraciones y 50 repeticiones de cada una.

5.3 Prueba online

Para complementar la experimentación offline descrita hasta aquí, se ha llevado a cabo una experiencia de evaluación comparativa con usuarios en línea con una selección de los sistemas de recomendación que han resultado más efectivos o representativos en las pruebas offline, y que resultan viables para una prueba en vivo. Uno de los objetivos de estas pruebas es contrastar y ver en qué medida las observaciones de la experimentación offline se confirman o matizan, dando un mejor acabado al estudio realizado.

Mientras las pruebas offline son sencillas de realizar una vez se dispone de una muestra de los datos objetivo, las pruebas online son mucho más difíciles, debido a que es necesario conseguir un número significativo de usuarios para que de ellas se puedan sacar conclusiones válidas. Aceptando este problema y contando con que la escala en número de usuarios será limitada, se ha planteado una aplicación para realizar pruebas con usuarios "en vivo".

En los siguientes apartados se detalla la descripción del proyecto, el desarrollo realizado para llevarlo a cabo, su funcionamiento concreto y los resultados obtenidos.

5.3.1 Descripción del proyecto

El proyecto consiste en crear una aplicación móvil que recomiende de forma personalizada usuarios en Twitter. Dicho de otra manera, se plantea desarrollar un sistema aplicación con la que los usuarios que tengan una cuenta en Twitter puedan recibir recomendaciones sobre los usuarios a quienes más pudiera interesarles seguir.

La idea inicial era realizar una comparación entre algunos de los mejores algoritmos del estudio offline para compararlos directamente con el algoritmo de *Who To Follow* de Twitter. Sin embargo, únicamente los usuarios pueden acceder a las recomendaciones recibidas por Twitter, de forma que es imposible obtenerla a través de la API y por tanto, realizar la comparación directa entre su sistema y los del presente estudio. Por ello, se ha decidido someter a estudio online una selección de los algoritmos estudiados para observar si los resultados online confirman los offline.

A continuación se describe el planteamiento del proyecto llevado a cabo, la tecnología que se ha decidido utilizar, un análisis breve de la competencia, y los objetivos y funcionalidad del sistema a desarrollar.

Planteamiento

En primer lugar, se ha decidido realizar una aplicación móvil debido a la gran explosión en años recientes del uso de los dispositivos móviles, principalmente smartphones y tablets. En la actualidad, la gran mayoría de personas cuentan con al menos un dispositivo móvil de uso personal que utilizan constantemente y tienen siempre a mano. Aprovechando esta ventaja, realizar una aplicación para estos dispositivos sería la mejor opción para llegar a la mayor cantidad de público posible, lo cual es una de las necesidades principales de las pruebas online.

Además, el objetivo final de las pruebas online es realizar una comparación entre un subconjunto reducido de los sistemas de recomendación analizados en las pruebas offline. A la hora de elegir los algoritmos a comparar, hay que tener en cuenta que no son viables aquellos que requieren la red completa para generar la recomendación a los usuarios, sino que únicamente se puede contar con lo que se llama la "red ego" de cada

usuario, que consiste en un grafo creado mediante la expansión iterativa de e followees hasta llegar a una profundidad máxima p, comenzando por expandir al usuario objetivo. En este caso, se ha tomado $e = 50$ y $p = 2$ para crear las redes ego de los usuarios. En la Figura 11 se muestra el conjunto de redes ego que han utilizado los algoritmos para recomendar a cada usuario.

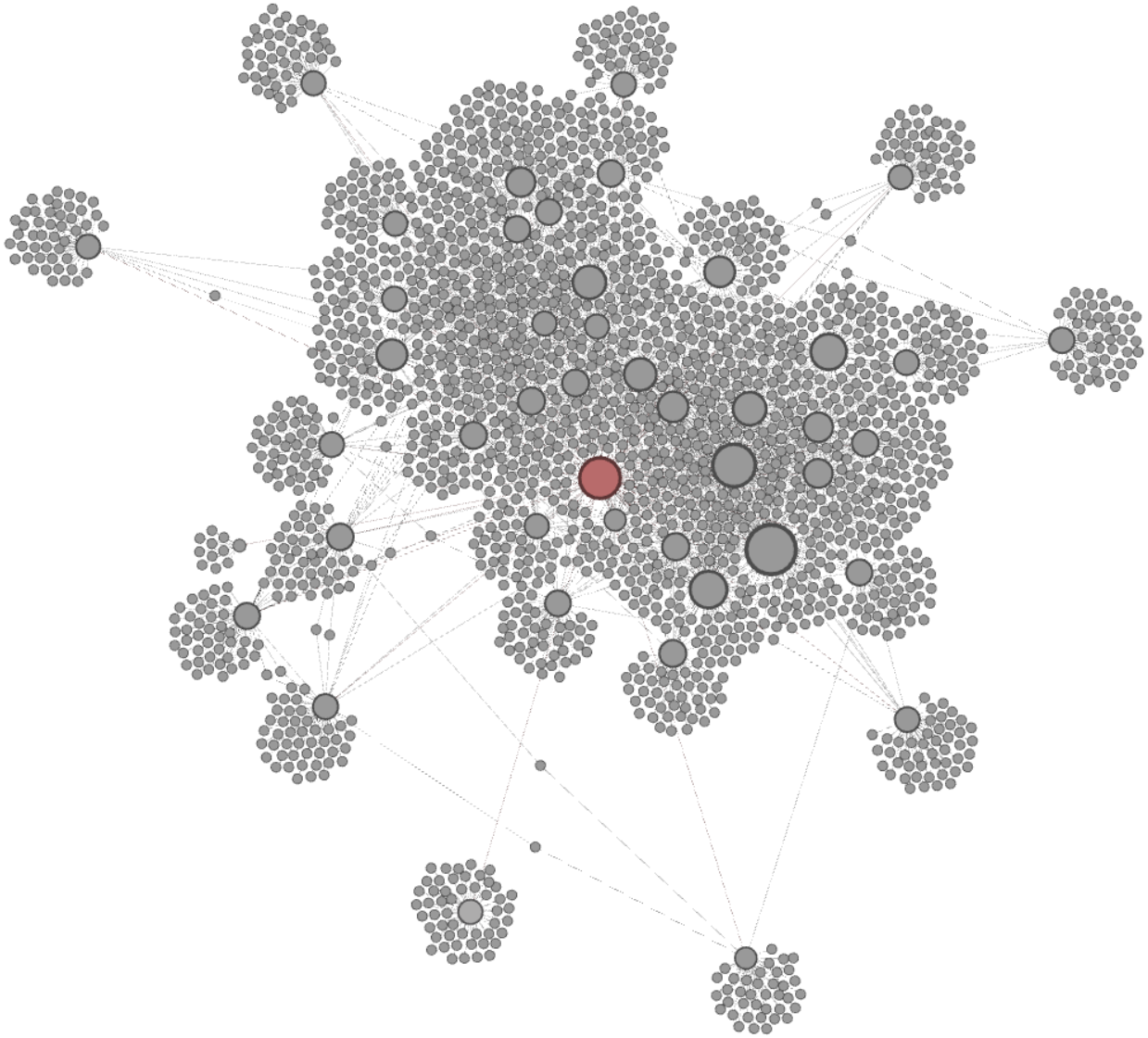

Figura 11. Ejemplo de red ego de un usuario (el nodo de color rojo).

Varios algoritmos utilizan la red completa para generar recomendaciones, pero en este caso sólo es viable disponer de la red ego del usuario objetivo. Aunque estos algoritmos podrían utilizar la red ego para realizar sus recomendaciones, se ha optado por no incluirlos en los experimentos online porque se entiende que estarían en desventaja frente a otros algoritmos que se basan en vecindarios. Así, los algoritmos seleccionados como objeto de estudio en este caso son cuatro:

- **Popularidad**: A partir de una selección de los 50 usuarios con más followers, este algoritmo recomienda por orden de popularidad aquellos usuarios que el usuario objetivo no sigue.

- **Adamic**: Busca recomendar usuarios similares al objetivo basándose en la importancia que tienen sus vecinos en común. En este caso, los usuarios que forman parte del vecindario común entre el usuario objetivo y el contacto candidato son aquellos usuarios seguidos por *u* que siguen a *v*. En cuanto a la importancia de los vecinos en común, se tiene en cuenta únicamente el número de followers de cada uno, de forma que cuantos más followers tiene más importante se considera.
- **ExtremeBM25**: En este caso se trata el grafo como no dirigido, y al igual que en Adamic se tienen en cuenta los vecinos en común entre el usuario objetivo y el candidato. La diferencia principal reside en que se presta atención a la importancia que tiene el enlace que los une con los vecinos, de forma que si un usuario en común a los dos tiene un grado muy alto, entonces se entiende que realmente no se tratan de enlaces con valor y apenas se tiene en cuenta. Se utiliza la versión Extreme de BM25 para poder reducir el número de parámetros a fijar.
- **QLJM**: Se trata de una recomendación realizada según la probabilidad de que el contacto candidato sea relevante para el usuario objetivo. Presta atención a los usuarios en común igual que los dos casos anteriores, pero se realiza un suavizado según la popularidad del vecino común.

Cada algoritmo genera su ranking de forma independiente, y a continuación, mediante un test AB (de cuatro fuentes en lugar de dos), se intercalan los resultados para formar un único ranking de resultados. Se ha seguido la técnica *team draft* (Radlinski 2008), que consiste en que en orden aleatorio y de cuatro en cuatro se elige uno de los recomendadores para colocar el siguiente contacto recomendado en el ranking final. Se finaliza al llegar en este caso hasta 20 contactos recomendados, incorporando cada uno 5 de sus recomendaciones en el ranking final.

Sin embargo, debido a que la limitación en el número de peticiones para obtener la red ego de enlaces tipo follow de un usuario es muy grande, como primera recomendación se realiza una basada en los retweets, que es mucho más rápida de conseguir. Se trata de un algoritmo que recupera los 200 últimos tweets del usuario objetivo, de esos selecciona los usuarios a los que le ha hecho retweet y les asocia un peso que se corresponde con el número de veces que ha habido ese tipo de interacción. Con estos datos se genera un grafo dirigido y ponderado, y se van expandiendo los nodos de la misma forma que con el usuario objetivo hasta llegar a una profundidad máxima (en este caso, 2). Para seleccionar los usuarios a recomendar, se utiliza la fórmula que se muestra a continuación:

$$f(u,v) = f(u,o) + \frac{weight(o,v)}{prof(o)+1}$$

Donde o es el predecesor del candidato v, $weight(o,d)$ es el peso del enlace entre los nodos (es decir, el número de tweets de v que o ha retwitteado), y $prof(o)$ es la profundidad del nodo predecesor.

Esta recomendación se complementa mediante un test AB con el recomendador de popularidad, de forma que en los casos extremos donde el usuario objetivo no ha realizado ningún retweet o no se haya conseguido el número suficiente de contactos a recomendar, se siga devolviendo un ranking de recomendaciones completo.

Selección de plataforma

De los numerosos sistemas operativos existentes destacan tres: iOS (Apple), Android (Google) y Windows Phone (Windows). Para el desarrollo de este proyecto se ha escogido la segunda opción, Android, por las razones que se muestran a continuación:

- **Número de usuarios**. Android es con diferencia el sistema operativo con uso más extendido en España (Figura 12), de forma que es más fácil llegar a la mayoría de la población utilizando este sistema operativo.

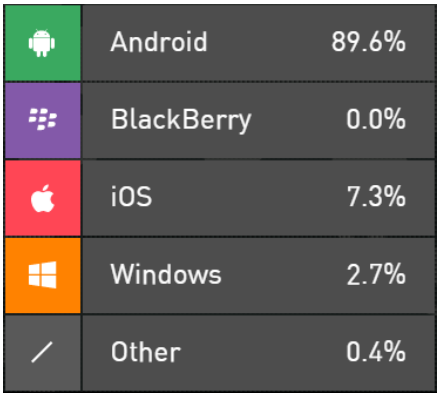

Figura 12. Proporción de uso de cada una de las tecnologías móviles en España.[16]

- **Aplicación de conocimientos adquiridos**. El desarrollo de aplicaciones en Android ha sido objeto de estudio (como una asignatura optativa) en la carrera y se ha aplicado (de forma opcional) en otra de máster. Realizando un proyecto completo e individual en este lenguaje sirve de asentamiento de los conocimientos adquiridos, así como de ampliación de los mismos al tener que solucionar problemas nuevos que pudiesen aparecer.
- **Facilidades para el desarrollo**. Una aplicación en Android puede programarse desde cualquier ordenador a diferencia de su competidor directo iOS, que obliga a realizarlo desde un ordenador Apple. Además, existe gran cantidad de documentación, guías y ejemplos a disposición pública, lo que facilita en gran manera el trabajo de los desarrolladores.
- **Facilidades para la publicación**. Para publicar una aplicación en Google Play para que las personas puedan descargársela y usarla en sus dispositivos, es necesario adquirir la licencia de publicación de 25$ USD. Sin embargo, al realizar el pago se obtienen varias ventajas, como el seguimiento exhaustivo de los progresos de la aplicación, consejos para su mejora y promoción, etc.

Es necesario decir que hay que ser cuidadoso con las versiones de Android que existen, ya que están en constante cambio y evolución. En este caso, se ha seleccionado como objetivo desarrollar la aplicación para aquella versión que cuente con más usuarios en el mercado actualmente, Android Kit Kat, debido principalmente a dos razones. Por un lado, se trata de una aplicación bastante sencilla, donde la visualización de datos no tiene complejidad alguna y existen muy pocos modelos de *layouts*. Por otro lado, si se utiliza un *layout* que no existía previamente, la información un móvil con una versión antigua no se visualiza de la misma forma y/o puede traer problemas de visualización).

[16] Fuente: http://www.kantarworldpanel.com/global/smartphone-os-market-share/ , octubre 2015.

Análisis de la competencia

Una vez se tiene una idea más o menos clara de lo que se quiere implementar, es necesario realizar un análisis de la competencia para ver si existe en el mercado una aplicación que realice la misma tarea o una similar. Si se encuentran aplicaciones iguales o similares, se pueden analizar para obtener las ventajas e inconvenientes de cada una para tenerlas en cuenta en el desarrollo de la aplicación propia, o incluso para que sirva como inspiración de nuevas ideas.

Realizando numerosas búsquedas en Google Play, no se ha conseguido encontrar ninguna aplicación que realice recomendaciones de usuarios que seguir en Twitter. La gran mayoría de aplicaciones relacionadas con Twitter se agrupaban en las siguientes categorías:

- **Cliente de Twitter**: realiza la misma función que la aplicación original de Twitter, pero intentando aplicarle mejoras por ejemplo, en la visualización de la información con nuevas combinaciones de colores, o en la funcionalidad añadiendo por ejemplo ver tweets por zonas. Ejemplos: Tweetr+, TwitPane, Bluejabb, Plume, Tweet Balloon, Tweet Lanes.
- **Gestión de contactos**: como su nombre indica, se trata de aplicaciones que permiten controlar los diferentes tipos de relaciones con usuarios, complementándolo con gráficas o notificaciones. Un usuario puede recibir una notificación cuando alguien comienza o deja de seguirle, así como ver de los usuarios a los que sigue quiénes no le siguen, de los que no sigue quiénes le siguen y qué usuarios le siguen y a la vez éste sigue. Ejemplos: Unfollowers, Twitpalas, TwTolls, Followers+, Track my Followers, Follow, Follow Tool, Catch The Follower!.
- **Gestión de cuenta**: parecida a la anterior, en este caso se trata de aplicaciones que permiten acciones a realizar desde la cuenta objetivo, como programar tweets para que se publiquen a una hora determinada, observar estadísticas, nuevos tipos de visualización de la información, etc. Ejemplos: Tuit Útil GOLD.
- **Búsqueda de seguidores**: son aplicaciones que mediante campañas o similares hacen que varios usuarios sigan la cuenta del usuario objetivo. En ninguno de los casos se trata de recomendaciones, sino que al descargar este tipo de aplicaciones se consiguen automáticamente nuevos seguidores, probablemente no de usuarios reales sino de cuentas robot. Ejemplos: Buscando amigos en Twitter, TwitGrow, Twitter Follow.

Al no encontrarse ninguna aplicación que realice la tarea objetivo de recomendación de usuarios en Twitter, se amplió la búsqueda fuera del contexto de Twitter para ampliarlo a cualquier otra red social. Sin embargo, los resultados obtenidos no fueron relevantes en ningún caso.

En conclusión, el análisis de la competencia en este caso ha servido únicamente para esencialmente obtener ideas de cómo presentar la información en la aplicación.

Objetivos y funcionalidad

El siguiente paso a realizar es una especificación de los objetivos y la funcionalidad que se quiere que la aplicación tenga.

El objetivo principal consiste en comparar los algoritmos de recomendación seleccionados según el número de aciertos que ha tenido cada uno, considerando como acierto que, a través de la aplicación, un usuario elija seguir a otro de los recomendados.

En cuanto a la funcionalidad de esta aplicación, se espera que los usuarios se identifiquen con sus credenciales de Twitter, que éstos reciban recomendaciones del sistema, puedan visitar el perfil de los usuarios recomendados, y que puedan seguir, dejar de seguir y eliminar de la recomendación a aquellos usuarios en los que tengan interés, lo hayan perdido o no les guste, respectivamente.

5.3.2 Desarrollo y funcionamiento

El sistema que forma la aplicación móvil no consta simplemente del proyecto Android desarrollado. Para la obtención de las recomendaciones para cada usuario, la aplicación realiza una petición HTTP a un servidor propio Apache (activo todo el tiempo) que recoge el identificador del usuario y devuelve en forma de JSON la lista de usuarios recomendados. Esta lista se puede obtener de dos maneras: si es la primera vez que el usuario usa la aplicación, el sistema genera en ese mismo momento una recomendación basada en sus retweets recientes, mientras que en el caso contrario basta con recoger el ranking correspondiente de la base de datos.

Existen además tres procesos en el servidor que están en background ejecutándose constantemente. Dos de ellos son programas que se encargan de coger un usuario que ya haya visto la última recomendación que se le había generado y le crea una nueva, obteniendo nuevamente su red ego y el ranking según los sistemas de recomendación seleccionados. El tercer proceso es un programa que se encarga de ver qué usuarios tienen una nueva recomendación preparada y twittea a través de la cuenta de Twitter creada para la aplicación (*@followppapp*) un texto en el que se le avisa al usuario objetivo (mediante una mención al mismo) de que tiene disponible una nueva recomendación preparada en la aplicación.

Como se puede intuir, la base de datos es el elemento clave del sistema, ya que mantiene el estado de todos los usuarios y sirve de conexión entre los procesos en background y el proceso principal de resolución de peticiones. Además, como se ha dicho anteriormente, la aplicación móvil se conecta directamente a la base de datos para almacenar todas las acciones que los usuarios realizan, de forma que está constantemente cambiando y siendo modificada.

Se trata entonces de un sistema formado por una aplicación en Android, un programa en Java y un conjunto de ficheros PHP para poder intercambiar información entre la aplicación y el servidor de forma sencilla, todas ellas incluyendo el uso de MySQL para la manipulación de datos de la base de datos.

El funcionamiento de la aplicación de forma completa se muestra en la Figura 13 y se describe de forma detallada a continuación de la misma.

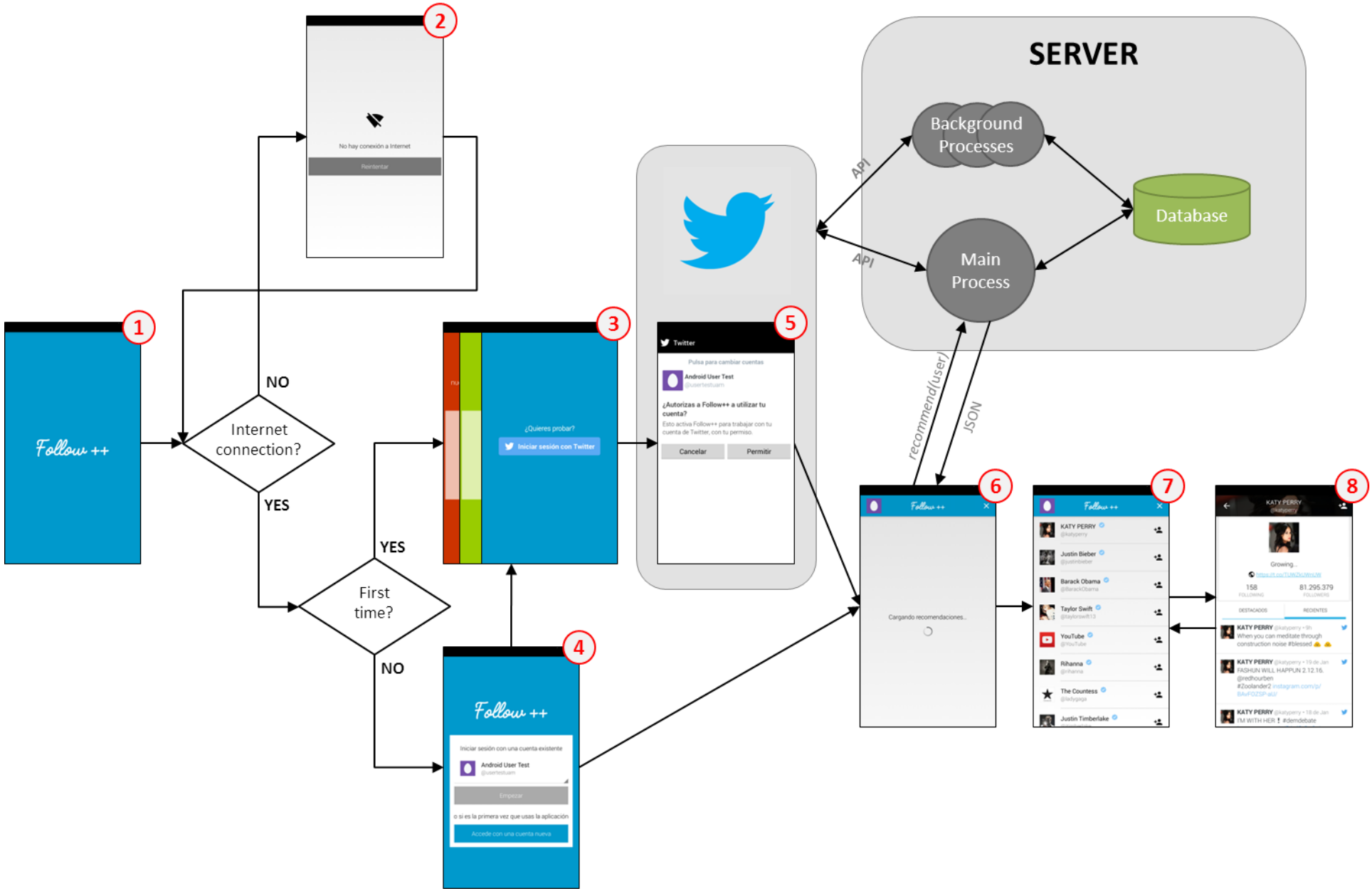

Figura 13. Diagrama de funcionamiento completo de la aplicación desarrollada Follow++.

1. Splashscreen: Al abrirse la aplicación se muestra el nombre de la misma junto con la animación de un pequeño pájaro. Se comprueba que el móvil tenga conexión a Internet y si ya existe alguna sesión con la que acceder.
2. Sin conexión: Si el móvil no tiene conexión a Internet, la aplicación no funcionaría correctamente, de forma que se muestra una pantalla de error. En esta pantalla hay un botón que permite intentar la reconexión a Internet, lo que permitiría continuar con la ejecución de la aplicación con normalidad.
3. Primer acceso y login mediante Twitter: Cuando se trata de la primera vez que el usuario accede a la aplicación, se muestra en un par de pantallas la motivación de la misma y se muestra un botón para iniciar sesión a través de Twitter. Se trata de una funcionalidad facilitada por la propia API de Twitter, y esto permite que los usuarios accedan directamente con las credenciales de su cuenta de Twitter, pidiéndoles además que acepten los términos de uso de la aplicación. Como resultado la aplicación obtiene un objeto en el que se muestra la información del usuario logueado, omitiendo datos como la contraseña, de forma que no es necesario almacenarla ni tomar medidas de seguridad para protegerla.
4. Acceso de antiguo usuario: Se trata de la pantalla de login, donde se puede seleccionar la sesión de usuarios que hayan accedido anteriormente o acceder con una cuenta nueva, lo cual llevaría a la pantalla de primer acceso y login mediante Twitter. En este caso no se pide nuevamente que se introduzcan las credenciales de usuario, ya que las sesiones una vez creadas se guardan en la base de datos de la aplicación del propio dispositivo móvil.
5. Aceptar condiciones: Al realizar satisfactoriamente el login mediante Twitter, se pide al usuario que autorice a la aplicación para acceder a los datos de la cuenta. Concretamente, la aplicación necesita tener accesos de lectura, para obtener la red ego del usuario, y de escritura, para que el usuario pueda seguir y dejar de seguir a usuarios desde la propia aplicación. Como es lógico, es estrictamente necesario que se dé la autorización para que la aplicación funcione y pueda realizar recomendaciones.
6. Cargar recomendaciones: Una vez se dispone de la sesión de un usuario válido, se realiza una petición HTTP al servidor para obtener la lista de recomendaciones generadas para el usuario objetivo. Si se trata de la primera vez que el usuario recibe una recomendación, el servidor genera un ranking basándose en los retweets que el usuario haya realizado, de forma que se obtenga de forma rápida (la limitación de peticiones de este tipo de información es baja) y no se tenga al usuario en espera. Si por el contrario no es la primera vez que el usuario accede a la aplicación, se muestra bien la última recomendación recibida anteriormente, o bien una nueva recomendación más compleja generada por los servidores con un mensaje de “nueva recomendación”.
7. Lista de recomendación: En cuanto a la visualización de los datos de las recomendaciones obtenidas, se muestra una lista de los usuarios recomendados ordenados de forma descendiente por su importancia estimada por los algoritmos de recomendación. A la derecha de cada elemento de la lista se encuentra un botón para comenzar a seguir un usuario, o para dejar de seguirlo si ya se había seguido. Asimismo, si se mantiene pulsado un elemento de la lista, aparece la opción de eliminarlo de la lista para que nunca más vuelva a ser recomendado por los sistemas de recomendación. Por último, en la parte derecha de la barra superior se encuentra una cruz, la cual permite cerrar la sesión del usuario logueado y volver a la pantalla de login.

8. Perfil de usuario: Al seleccionar un usuario recomendado, se muestra su perfil de usuario de Twitter (imagen de perfil, descripción, número de seguidores, etc.). Como extra, se muestra también los tweets recientes y los tweets más destacados del usuario, de forma que el usuario objetivo pueda observar su actividad y decidir con mejor criterio si le interesa o no seguir a esa persona. Además, en la parte derecha de la barra superior se muestra el icono para hacer *follow/unfollow* igual que en la pantalla de la lista de recomendación, de forma que en esta misma pantalla el usuario pueda seguir o dejar de seguir al usuario cuyo perfil está explorando.

Además de almacenar en el servidor de forma detallada cada recomendación que se realiza (momento en que se genera, contactos recomendados, su posición en el ránking, si el usuario lo sigue, identificador del algoritmo que propuso la recomendación de ese contacto, etc.), todas las interacciones que realizan los usuarios con la aplicación se envían y guardan en el mismo. Así, tiene información de las siguientes acciones:

- Cuando el usuario sigue a un contacto recomendado. Se guarda el momento, el usuario objetivo y el contacto seguido.
- Cuando el usuario deja de seguir a un contacto recomendado que había seguido anteriormente. Se guarda el momento, el usuario objetivo y el contacto que éste ha dejado de seguir.
- Cuando el usuario accede al perfil de un contacto recomendado. Se guarda el momento, el usuario objetivo y el contacto explorado.
- Cuando el usuario sale del perfil de un contacto recomendado al que había accedido anteriormente. Se guarda el momento, el usuario objetivo y el contacto que ha terminado de explorarse.

La aplicación final se llama Follow++ y se encuentra disponible en Google Play. Se puede descargar desde esta URL[17] o bien escaneando el código QR siguiente con el dispositivo móvil (Figura 14):

Figura 14. Código QR que enlaza a la página de Follow++ en Google Play.

5.3.3 Resultados obtenidos

El 26 de octubre de 2015, la aplicación Follow++ desarrollada en el presente trabajo fue subida a la plataforma de Google Play para que cualquier persona con dispositivo Android pudiese descargarla. Hasta el momento (finales de enero de 2016), se ha conseguido un total de 20 usuarios, que desafortunadamente no es un número suficiente de usuarios para que los resultados obtenidos sean completamente válidos. Sin embargo, a continuación se muestra y analiza los resultados extraídos de estas interacciones para que, aunque no sean fiables, se pueda observar las posibles tendencias.

Por un lado, con la plataforma de Google Play Developer Console (accesible por disponer de la licencia para subir aplicaciones a Google Play), se puede realizar el se-

[17] Enlace: https://play.google.com/store/apps/details?id=ir.eps.uam.es.twitter

guimiento de la aplicación. De esta forma, por ejemplo, se puede obtener el número de instalaciones totales por usuario (incluyendo los que posteriormente la desinstalaron) a lo largo de los meses (Figura 15).

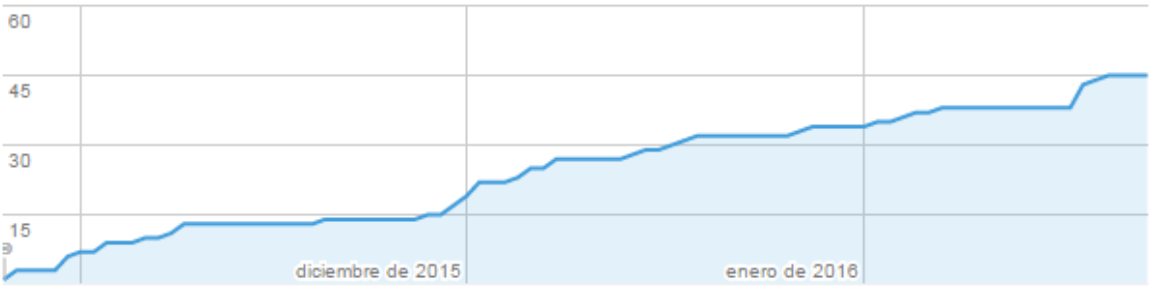

Figura 15. Progreso del número de instalaciones de la aplicación Follow++.

Desde la publicación de la aplicación, hasta 45 usuarios se descargaron la aplicación, pero unos 25 la desinstalaron sin llegar siquiera a loguearse (ya que no figuran en la base de datos). Lo más probable es que estos usuarios hayan rechazado darle los permisos a la aplicación de lectura (para acceder a sus followers y followees) y de escritura (para realizar follows y unfollows), posiblemente por cuestiones de desconfianza.

Otro dato interesante que se puede obtener de Google Play Developer Console es la versión Android que utilizan los usuarios de la aplicación. En la Figura 16 se muestra a la izquierda las proporciones de las versiones de Android que han descargado la aplicación, y en la derecha, estas mismas proporciones pero a nivel global para el conjunto de aplicaciones de la categoría de sociedades (a la que la aplicación propia pertenece).

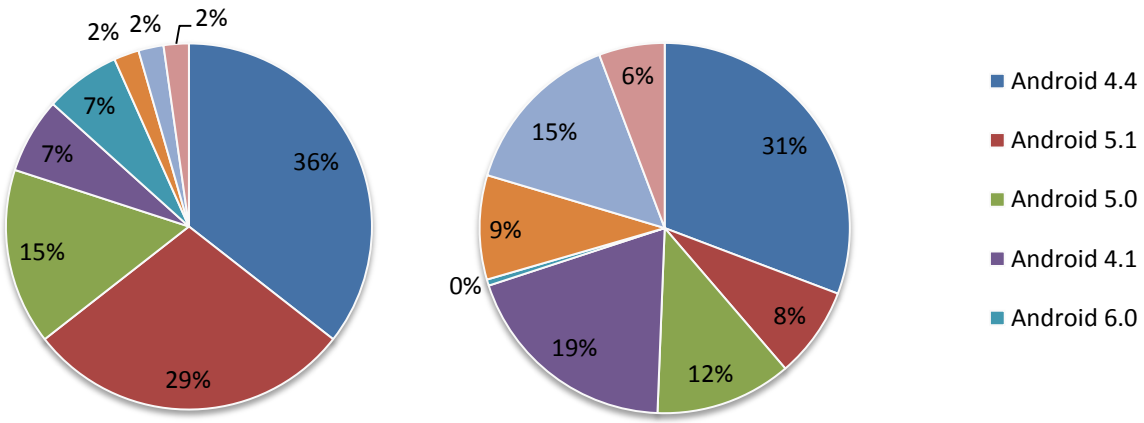

Figura 16. Proporciones de las versiones de Android utilizadas en la aplicación de Follow++ (izquierda) y de forma global en las aplicaciones del mismo tipo (derecha).

Como se puede observar, las proporciones no se mantienen del todo, pero la versión de Android a la que va dirigida la aplicación ocupa la mayor proporción de todas, tanto en los usuarios de la aplicación como a nivel global. Como extra, es de destacar que actualmente la última versión de Android tiene una presencia mucho menor que el resto de versiones, por lo que ha sido un acierto orientar la visualización de datos para las versiones más utilizadas en lugar de las más modernas.

Por otro lado, todas las interacciones que los usuarios han tenido con la aplicación se han almacenado en la base de datos como se ha dicho anteriormente. De la base de datos entonces se puede sacar mucha información valiosa para principalmente comparar

la calidad de los algoritmos de recomendación utilizados, aunque antes hay que realizar un análisis de los usuarios que han utilizado la aplicación.

Así, en primer lugar la Figura 17 muestra el número de recomendaciones que ha recibido cada usuario. Como se puede observar, únicamente la mitad de los usuarios han utilizado la aplicación más de una vez, ya que aquellos que tienen dos recomendaciones tienen preparada la segunda recomendación (con los algoritmos a comparar) pero no la han llegado a ver (sino tendrían la siguiente, la tercera, preparada).

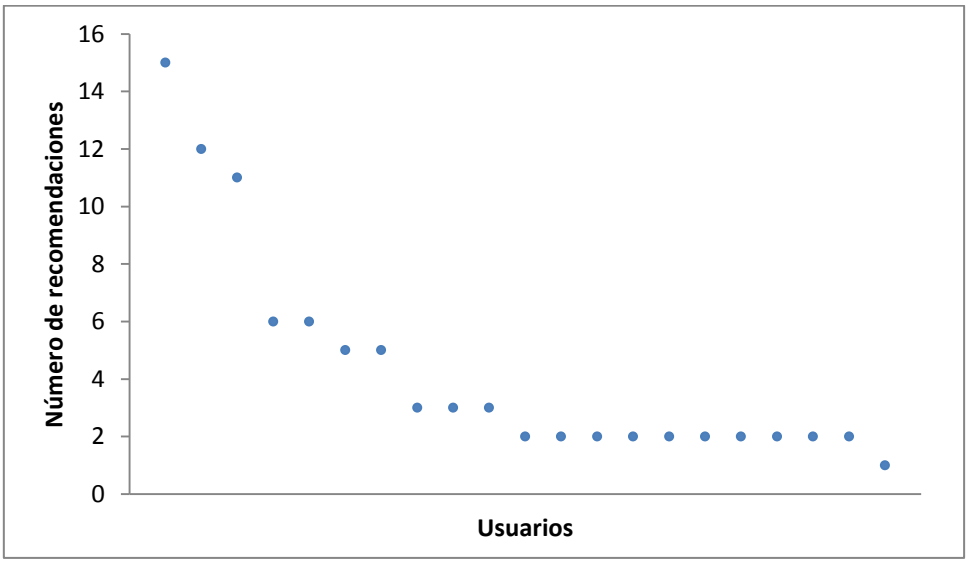

Figura 17. Número de recomendaciones que han recibido los usuarios.

En segundo lugar, analizamos los aciertos que las recomendaciones han tenido, es decir, los nuevos enlaces follow que el usuario ha establecido con los usuarios que se le han recomendado. En la Figura 18 se observa el número de aciertos que se han obtenido en cada posición.

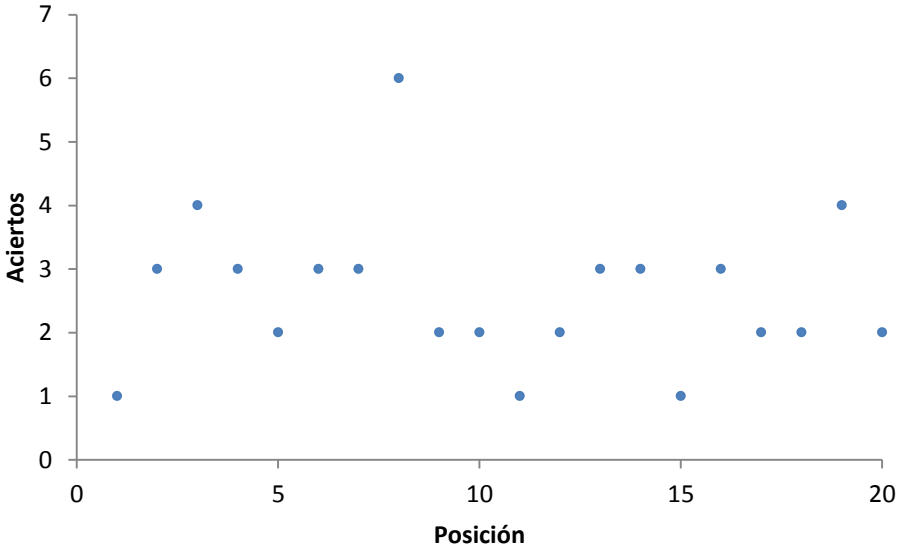

Figura 18. Número de aciertos obtenidos en cada posición.

Normalmente, cuando se observa el número de aciertos observados por posición se suele obtener una gráfica que desciende rápidamente con la posición, de forma que los primeros elementos suelen ser los que típicamente reciben más atención por parte de los

usuarios y los últimos los que menos (bien porque el usuario no llega a verlos o bien porque ya ha encontrado el usuario lo que buscaba en elementos anteriores). Sin embargo, en este caso todas las posiciones reciben un número de aciertos similar, en torno a 2 y 3, y esto es debido a varias razones. Por un lado, el usuario recibe las recomendaciones y se encuentra en modo *browsing*, es decir, sin ningún objetivo concreto (como encontrar el resultado de una búsqueda), y por ello se interesa en mirar todas las recomendaciones recibidas. Por otro lado, se trata de recomendaciones pequeñas, de máximo 20 elementos, que además se visualizan rápidamente, por lo que el usuario apenas tiene que gastar esfuerzo en explorar todas las recomendaciones recibidas. Además, las recomendaciones están formadas por un test AB de cuatro recomendadores, de forma que lo que se está recomendando al final es el top 5 de cada algoritmo, lo cual da una probabilidad de acierto muy alta para cada recomendador.

Es necesario decir que al observar el acierto no se ha tenido en cuenta los unfollows que se han realizado en la aplicación, ya que se tratan de casos aislados (únicamente 3 unfollows frente a 55 follows) y omitirlos en el estudio apenas cambiaría los resultados obtenidos.

En tercer lugar, prestando atención esta vez a cuando el usuario entraba y salía de los perfiles de los contactos recomendados, se puede observar el tiempo que los usuarios han pasado explorando los perfiles. En la Figura 19 se muestra el tiempo (en segundos) que han estado los usuarios explorando perfiles.

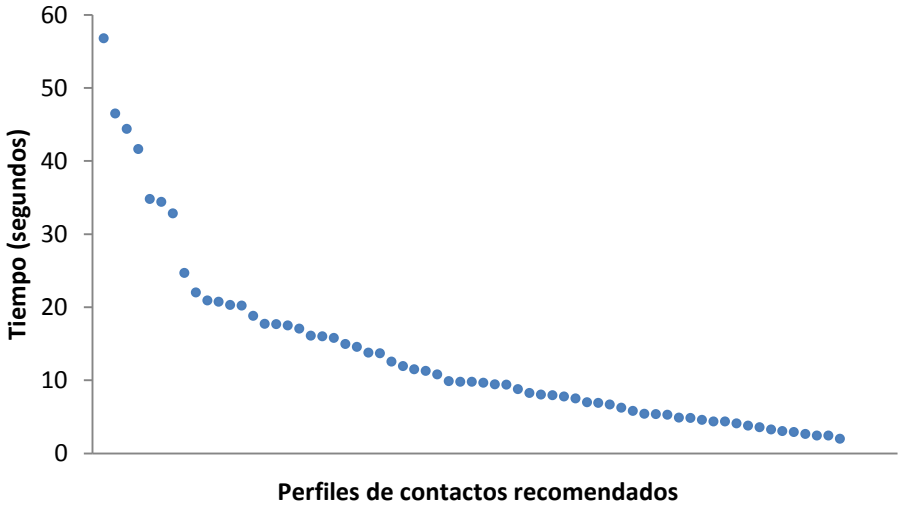

Figura 19. Tiempo que los usuarios han dedicado a explorar perfiles (los datos están ordenados de mayor a menor tiempo de inspección por el usuario).

En total se han realizado en torno a 65 exploraciones de perfil, de forma que de media a cada usuario le ha interesado conocer el perfil de en torno a 3 usuarios recomendados, y ha pasado además algo menos de 15 segundos explorando su perfil.

Por último, es objetivo de estudio principal la comparación entre los algoritmos utilizados para la creación de las recomendaciones a los usuarios de la aplicación. Esta comparación se puede realizar prestando atención a tres características: número de aciertos, de fallos (contactos recomendados que los usuarios han eliminado de la lista de recomendación) y de visitas al perfil del contacto recomendado. Así, el mejor algoritmo será el que maximice el primero y el tercero, y además minimice el segundo. La cantidad de aciertos, fallos y visitas obtenidas por cada recomendador se muestran todas a continuación en la Figura 20.

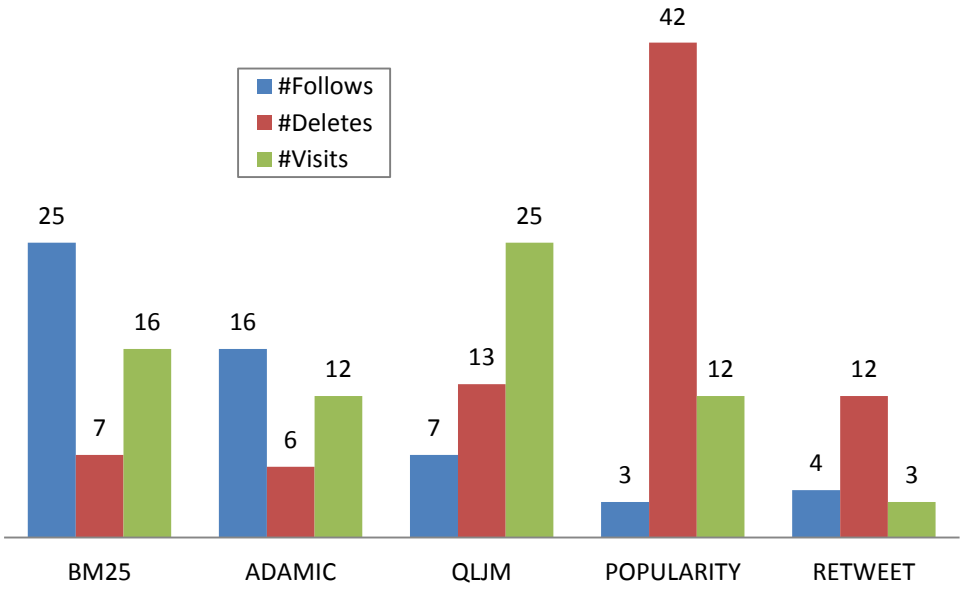

Figura 20. Cantidad de aciertos (azul), fallos (rojo) y visitas (verde) obtenidas por cada uno de los recomendadores utilizados en la aplicación.

En cuanto a los aciertos, se observa que BM25 destaca en gran medida sobre el resto, ya que casi la mitad de los aciertos han sido de usuarios que este algoritmo ha recomendado. En segundo y tercer lugar se encuentran Adamic y QLJM respectivamente, alejándose mucho de los algoritmos de Retweet y el de Popularidad, cuyo acierto es con diferencia el peor.

En el caso de los fallos, se obtienen los resultados opuestos al caso anterior, que es lo efectivamente se esperaría que ocurriese. El algoritmo de Popularidad es peor, ya que cuenta con más de la mitad de los fallos de recomendación. Los siguientes peores son QLJM y Retweet, que se alejan mucho de Popularidad pero tienen casi el doble de fallos que Adamic y BM25.

Finalmente, se analiza el interés que despiertan en los usuarios las recomendaciones de los algoritmos y que los lleva a explorar su perfil. En este caso, se observa un comportamiento totalmente diferente a los dos anteriores. QLJM es el que más visitas ha generado, a éste le siguen muy cerca BM25, Adamic y Popularidad, y en el último lugar con una diferencia notable está el algoritmo de Retweet.

El número de visitas que un usuario realiza sobre los contactos recomendados se puede interpretar como la novedad que aportan los algoritmos al usuario, de forma que aunque por ejemplo QLJM no es el mejor en maximizar el acierto o minimizar los fallos, puede ser interesante utilizarlo en el caso en el que se busca aportar novedad al usuario.

En conclusión, de forma general (y entendiendo la limitación de la validez de las conclusiones a sacar debido a tener tan pocos usuarios) se puede observar que la Popularidad es en realidad el peor, ya que tiene muchos fallos y pocos aciertos. En cuanto a la novedad que aporta este sistema, se esperaría que fuese muy baja, ya que se trata de un algoritmo que recomienda a las personas más populares de Twitter. Sin embargo, existe un matiz: este algoritmo recomienda las personas con más followers de Twitter a nivel mundial, de forma que muchos de ellos son personas famosas principalmente en Estados Unidos y en España (de donde son la mayoría de usuarios de la aplicación), no se conocen.

Es de destacar el buen funcionamiento del algoritmo de recomendación de BM25 adaptado en el presente trabajo del contexto de IR. Es el sistema que mejores resultados ha obtenido: por un lado, es el que más aciertos ha tenido con diferencia, es uno de los que menos fallos ha tenido (siendo superado sólo por Adamic), y además es el segundo algoritmo que más novedad aporta al usuario. De esta forma, este sistema de recomendación consigue muy buenos resultados en varios de los ámbitos a los que se le ha sometido, por lo que se puede considerar que es el mejor de los probados.

Por último, se puede decir que los resultados obtenidos en las pruebas online confirman en gran medida lo observado en las pruebas online, a pesar de las limitaciones de estas últimas. De los seleccionados en estas pruebas, BM25 resultó ser el que mejor acierto conseguía en las pruebas offline, seguido por QLJM y Adamic, mientras que en cuestiones de novedad estaba ligeramente por debajo de QLJM.

6. Conclusiones

6.1 Resumen y contribuciones

En este trabajo se ha abordado el problema de la recomendación de usuarios en las redes sociales, incidiendo en la definición de algoritmos de recomendación, su evaluación, y el análisis de los diferentes puntos de vista que pueden definir lo que significa una buena recomendación. A este respecto planteamos valorar la calidad de las recomendaciones no sólo por el acierto sino también por dimensiones como la novedad y la diversidad. Además, se aporta un nuevo punto de vista que consiste en no sólo valorar la aportación individual de la recomendación a cada usuario visto aisladamente, sino considerar la mejora de ciertas características de la red social a las que un algoritmo de recomendación puede contribuir. El trabajo realizado tiene una vertiente principal centrada en labores de investigación, que se complementa con una segunda vertiente más orientada a la realización de un producto para poner en práctica lo analizado en la investigación.

La investigación arranca con una comparación entre diferentes grupos de sistemas de algoritmos: los especializados en la recomendación en redes sociales, los tradicionales utilizados típicamente para la recomendación de ítems, y algoritmos adaptados de otro contexto, la Recuperación de Información. El objetivo principal es ver en qué grado mejoran los algoritmos especializados en el contexto frente a los tradicionales, y observar si adaptaciones de IR que nunca se habían realizado podían llegar a ser competitivas en este nuevo ámbito.

Para realizar esto, en primer lugar se ha estudiado la literatura relacionada con la recomendación en el contexto específico de las redes sociales, así como de otras tareas relacionadas como lo es la predicción de links en un grafo. De este análisis se han extraído algunos de los algoritmos destacados para la recomendación de contactos en redes sociales, los cuales han sido implementados, al igual que algoritmos tradicionales y adaptaciones de IR, para la comparación con el resto. Además, se han realizado pequeños análisis para optimizar algunos de los algoritmos de recomendación, como BM25 y PageRank personalizado.

En segundo lugar, para realizar las recomendaciones es necesario disponer de una muestra de la red social objetivo para realizar las pruebas. En este caso, se ha utilizado una muestra de la red social de Twitter que cuenta con poco más de 10.000 usuarios y todos los enlaces existentes entre ellos. La partición realizada es temporal y además es pura, es decir, que la parte de entrenamiento fue formada primero tomando la información de los usuarios hasta una fecha determinada, y los datos de test se recopilaron meses después sobre la actividad de los mismos usuarios a partir de la fecha de finalización del primer período de datos.

Por último, nos hemos centrado en analizar la calidad de las recomendaciones realizadas. Se han tenido en cuenta métricas tradicionales que miden el acierto (cuántos contactos de test está prediciendo correctamente la recomendación), otras más innovadoras que miden la novedad (qué tan nuevo es cada contacto recomendado para el usuario) y la diversidad (qué tan diferentes son entre sí los contactos recomendados al usuario). Asimismo, se propone y desarrolla una nueva perspectiva de métricas de evaluación que

representan nociones de social welfare, que valoran en qué medida los enlaces entre los contactos recomendados y el usuario objetivo mejoran características de la red, tales como la diversidad estructural.

Al realizar la comparación desde estos diferentes puntos de vista, el algoritmo que ha resultado ser más efectivo en conjunto en las diferentes dimensiones es HKV, representante de los algoritmos de filtrado colaborativo basado en factorización de resultados. No está documentada la adaptación de este algoritmo a la recomendación de contactos, y es parte de los hallazgos del presente trabajo encontrar que tiene un comportamiento muy efectivo y equilibrado en esta tarea. Por otra parte, la adaptación aquí propuesta de algoritmos de IR ha dado lugar a resultados igualmente muy competitivos en cuanto al acierto y diversidad, si bien en las métricas de diversidad estructural no se obtienen resultados óptimos.

Además, se han resuelto algunas de las dudas que surgen al estar en el contexto de las redes sociales, como si nos define mejor la gente que nos sigue o la gente a la que seguimos, donde se ha observado que en la mayoría de algoritmos se obtienen mejores resultados tomando la gente a la que el usuario sigue como vecindario útil.

A partir de estos resultados, se ha investigado la reordenación de recomendaciones para mejorar la diversidad y las métricas de diversidad estructural manteniendo un nivel de acierto razonable. Así, se ha seleccionado la recomendación de HKV como objetivo, y se ha sometido a comparación los algoritmos de reranking de XQuAD y Gini, que mejoran la diversidad, el de Infomap, que optimiza la modularidad, y se ha creado un nuevo reranking híbrido en el que se busca mejorar la modularidad sólo si favorece a la diversidad. Los resultados validan el efecto coherente de la estrategia: cada reordenación logra optimizar su objetivo(s) específico(s) en el sentido de obtener el mejor resultado frente a todas las demás alternativas de la batería de pruebas.

Por otra parte, y como vía de valoración de la utilidad de las métricas de diversidad, hemos estudiado la forma que cada uno de estos rerankers fomentan la propagación de información. Hemos investigado este aspecto simulando la propagación de tweets en diferentes versiones de la red social (cada una añadiendo enlaces recomendados por un algoritmo diferente) y observando cómo evoluciona la diversidad de la información recibida por los usuarios, medida de diferentes maneras. Los experimentos han confirmado el efecto que la diversidad estructural (fomentada por un reranker orientado a la misma) tiene en la diversidad de información (medida en términos de diversidad de contenido, basada en hashtags).

Complementando las pruebas offline, se ha desarrollado una aplicación móvil en Android para valorar en qué medida los resultados obtenidos en la investigación se confirman con usuarios que reciben de verdad las recomendaciones y ante un contexto que plantea requisitos más restrictivos en términos de tiempo de respuesta (por lo cual, por ejemplo, es inviable contar con una red de usuarios tan amplia con la utilizada en las pruebas offline). Se han seleccionado para ello cuatro de los algoritmos analizados y se ha observado que de forma general se obtienen resultados muy similares a los de las pruebas offline. La escala de la prueba online es más limitada en escala, por lo que estos resultados no se pueden considerar como plenamente concluyentes, pero sí hemos podido validar técnicamente la aplicabilidad de nuestros métodos, y obtener evidencia preliminar adicional de su efectividad comparativa.

6.2 Trabajo futuro

Son múltiples las líneas de continuación del trabajo realizado hasta este punto. En primer lugar, si bien en este trabajo se ha implementado una amplia gama de algoritmos de recomendación abarcando los que hemos juzgado mejores y/o más representativos, es posible naturalmente ampliar la batería de algoritmos, incluyendo por ejemplo alguno más de factorización de matrices. De la misma forma, vemos amplias posibilidades de continuación y extensión de la perspectiva del bien social planteada en este trabajo, así como de los efectos que la diversidad estructural puede tener en la información que se propaga por la red. A este respecto consideramos que en este trabajo hemos dado un primer paso que da pie a un amplio camino potencial por recorrer.

Por citar una cuestión con mucho recorrido potencial de investigación futura en este sentido, el presente trabajo suscita implícitamente la pregunta de en qué medida la diversidad estructural de una red social es algo beneficioso. Nuestra respuesta, parcial, se ha encaminado a relacionar la diversidad de estructura con su efecto en la diversidad de la información que circula por la red. Pero pueden sin duda investigarse y valorarse las ventajas de la diversidad estructural desde más puntos de vista, que exceden el alcance del presente trabajo. Lo mismo podría decirse de la diversidad de información: ¿es deseable o no? Naturalmente no existe una única respuesta, pero es claro que un equilibrio entre diversidad y especialización es positivo en general, evitando por ejemplo el efecto de la llamada “filter bubble” (Pariser 2012).

Entrando en un plano más técnico, en el momento en el que se ha realizado este trabajo no existe la posibilidad de obtener la recomendación que el sistema de Who To Follow de Twitter realiza a los usuarios (actualmente sólo el usuario objetivo puede ver las recomendaciones de Twitter y es imposible obtenerlas mediante la API). Sin embargo, es posible que en un futuro esto cambie, y en ese caso sería muy interesante someter a comparación las recomendaciones de Twitter con las de algunos de los algoritmos que mejores resultados han obtenido en este estudio.

Por último, a raíz de los resultados obtenidos con la reordenación de recomendaciones, sería interesante continuar indagando en las mejoras que estos algoritmos pueden aportar a las recomendaciones. Por ejemplo, de la misma forma que se ha creado el algoritmo que optimiza simultáneamente la modularidad y la diversidad de Gini, a éste se puede añadir una tercera dimensión para mejorar en alguna otra característica del grafo o aportar una nueva perspectiva, como la novedad de la recomendación. Asimismo, puede ser interesante someter tanto el *baseline* como los diferentes rerankers a pruebas online, donde se pueda obtener el feedback directo de usuarios y observar las diferencias entre los algoritmos.

Por otra parte, los experimentos del presente trabajo se han realizado sobre el grafo de follows de Twitter, utilizando vínculos de seguimiento tanto como datos de entrada para los algoritmos de recomendación, como datos de test para evaluarlos. Pero cabe ampliar los experimentos utilizando vínculos de interacción (es decir retweets, replies, etc.) entre usuarios (por ejemplo, se considera que existe un enlace de un usuario u a otro v si u ha hecho retweet de un tweet publicado por v). A la fecha en que se escribe esta memoria, se ha obtenido ya una nueva muestra de la red social de Twitter utilizando los enlaces de interacción en lugar de los de follow. Con esto, se busca volver a comparar los algoritmos de recomendación con una red de naturaleza ligeramente diferente y observar si los que mejor resultado daban con la red de follows se mantienen en la de interacciones. Por otro lado, se han realizado pruebas sobre otros conjuntos de

datos de redes sociales, tanto dirigidas como simétricas, para probar la consistencia en los resultados de los algoritmos en diferentes contextos. En resultados preliminares que se están obteniendo a fecha presente se observa una dependencia importante de los resultados con las características de los conjuntos de datos (densidad, modularidad, coeficiente de clustering, etc.), lo que motiva ampliar la investigación para identificar y entender los factores de los datos que intervienen en las diferencias de rendimiento de los algoritmos.

Referencias

L. A. Adamic, E. Adar. Friends and Neighbors on the Web. Social Networks Journal, 25(3), July 2003, pp. 211-230.

G. Adomavicius, A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), June 2005, pp. 734-749.

A. Alhambra. Recomendación de contactos en Twitter. Trabajo Fin de Grado, Escuela Politécnica Superior, Universidad Autónoma de Madrid, June 2014.

J. Aranda, I. Givoni, J. Handcock, D. Tarlow. An online social network-based recommendation system. Technical report (pending results). 2007.

R. Baeza-Yates, B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.

R. Baeza-Yates, B. Ribeiro-Neto. Modern Information Retrieval - the concepts and technology behind search, 2nd Edition. Pearson Education Ltd., 2011.

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), October 2008.

P. Bonhard, M. A. Sasse. 'Knowing me, knowing you' - Using profiles and social networking to improve recommender systems. BT Technology Journal, 24(3), July 2006, pp. 84-98.

P. Castells, N. J. Hurley, S. Vargas. Novelty and Diversity in Recommender Systems. In Francesco Ricci, Lior Rokach, Bracha Shapira (Eds.): Recommender Systems Handbook, 2nd edition. Springer, 2015, pp. 881-918.

O. Chapelle, S. Ji, C. Liao, E. Velipasaoglu, L. Lai, S. L. Wu. Intent-based diversification of web search results: Metrics and algorithms. Information Retrieval Journal, 14(6), December 2011, pp. 572-592.

C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, I. MacKinnon. Novelty and Diversity in Information Retrieval Evaluation. 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008). Singapore, July 2008, pp. 659-666.

M. Clements, A. P. De Vries, M. J. Reinders. The task-dependent effect of tags and ratings on social media access. ACM Transactions on Information Systems (TOIS), 28(4). November 2010, Article 21.

P. De Meo, E. Ferrara, G. Fiumara, A. Provetti. On Facebook, most ties are weak. Communications of the ACM 57(11), November 2014, pp. 78-84.

S. V. Dongen. Graph Clustering by Flow Simulation. University of Utrecht, Holanda, May 2000.

D. Goldberg, D. Nichols, B. M. Oki, D. Terry. Using collaborative filtering to weave an information tapestry. Communications of the ACM 35(12), December 1992, pp. 61-70.

S. A. Golder, S. Yardi, A. Marwick, D. Boyd. A Structural Approach to Contact Recommendations in Online Social Networks. Workshop on Search in Social Media (SSM). Boston, MA, USA, July 2009, pp. 1-4.

M. S. Granovetter. The strength of weak ties. American Journal of Sociology, 78(6), May 1973, pp. 1360-1380.

P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, R. Zadeh. WTF: The Who to Follow Service at Twitter. 22nd International Conference on World Wide Web (WWW 2013). Rio de Janeiro, Brazil, May 2013, pp. 505-514.

J. Hannon, M. Bennet, B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. 4th ACM Conference on Recommender Systems (RecSys 2010). Barcelona, Spain, September 2010, pp. 199-206.

T. Hofmann. Probabilistic latent semantic indexing. 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999). Berkeley, CA, August 1999, pp. 50-57.

Y. Hu, Y. Koren, C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. 8th IEEE International Conference on Data Mining (ICDM 2008). Pisa, Italy, December 2008, pp. 263-272.

X. L. Huang, M. Tiwari, S. Shah. Structural Diversity in Social Recommender Systems. 5th ACM RecSys Workshop on Recommender Systems and the Social Web (RSWeb 2013). Hong Kong, China, October 2013.

D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. Institute of Radio Engineers, 40(9), September 1952, pp. 1098-1101.

K. Järvelin, J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000). Athens, Greece, July 2000, pp. 41-48.

H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, K. Tsuda. Link propagation: A fast semi-supervised learning algorithm for link prediction. SIAM International Conference on Data Mining (SDM 2009). Sparks, NV, USA, April 2009, pp. 1100-1111.

Y. Kim, K. Shim. TWITOBI: A Recommendation System for Twitter Using Probabilistic Modeling. 11th IEEE International Conference on Data Mining (ICDM 2011). Vancouver, Canada, December 2011, pp. 340-349.

I. Konstas, V. Stathopoulos, J. M. Jose. On Social Networks and Collaborative Recommendation. 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009). Boston, MA, USA, July 2009, pp. 195-202.

Y. Koren, R. Bell, C. Volinsky. Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42(8), August 2009, pp. 30-37.

S. M. Kywe, E. P. Lim, F. Zhu. A Survey of Recommender Systems in Twitter. 4th International Conference on Social Informatics (SocInfo 2012). Lausanne, Switzerland, December 2012, pp. 420-433.

J. H. Lee. Analyses of Multiple Evidence Combination. 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1997). Philadelphia, PA, USA, July 1997, pp. 267-276.

R. Lempel, S. Moran. SALSA: The Stochastic Approach for Link-Structure Analysis. ACM Transactions on Information Systems (TOIS). April 2001, 19(2), pp. 131-160.

D. Liben-Nowell, J. Kleinberg. The link prediction problem for social networks. 12th ACM International Conference on Information and Knowledge Management (CIKM 2003). New Orleans, LA, USA, November 2003, pp. 556-559.

R. N. Lichtenwalter, J. T. Lussier, N. V. Chawla. New perspectives and methods in link prediction. 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010). Washington, DC, USA, July 2010, pp. 243-252.

F. Liu, H. J. Lee. Use of a social network information to enhance collaborative filtering performance. Expert Systems with Applications. Elsevier, 37(7), July 2010, pp. 4772-4778.

H. Ma, H. Yang, M. R. Lyu, I. King. SoRec: Social Recommendation Using Probabilistic Matrix Factorization. 17th ACM Conference on Information and Knowledge Management (CIKM 2008). Napa Valley, CA, USA, October 2008, pp. 931-940.

J. Moody, D. R. White. Structural Cohesion and Embeddedness: A Hierarchical Concept of Social Groups. American Sociological Review, 68(1), February 2003, pp. 103-127.

M. E. Newman. Finding community structure in networks using the eigenvectors of matrices. Physical review E, 74(3), May 2006.

M. E. Newman, M. Girvan. Finding and evaluating community structure in networks. Physical review E, 69(2), August 2004.

G. K. Orman, V. Labatut, H. Cherifi. On Accuracy of Community Structure Discovery Algorithms. Journal of Convergence Information Technology, 6(11), December 2011, pp. 283-292.

L. Page, S. Brin. The Anatomy of a Large-Scale Hypertextual Web Search Engine. 7th International World Wide Web (WWW 1998). Brisbane, Australia, April 1998.

Eli Pariser. The Filter Bubble: How the New Personalized Web Is Changing What We Read and How We Think. Penguin Books, 2012..

J. M. Ponte, W. B. Croft. A language modeling approach to information retrieval. 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998). Melbourne, Australia, August 1998, pp. 275-281.

R. Rabbany, M. Takaffoli, J. Fagnan, O. R. Zaiane, R. J. G. B. Campello. Communities Validity: Methodical Evaluation of Community Mining Algorithms. Social Network Analysis and Mining Journal, 3(4), August 2013, pp. 1039-1062.

F. Radlinski, M. Kurup, T. Joachims. How does clickthrough data reflect retrieval quality?. 17th ACM Conference on Information and Knowledge Management (CIKM 2008). Napa Valley, CA, USA, October 2008, pp. 43-52.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. ACM Conference on Computer Supported Cooperative Work (CSCW 1994). Chapel Hill, NC, October 1994, pp. 175-186.

F. Ricci, L. Rokach, L., B. Shapira, P. B. Kantor (Eds.). Recommender Systems Handbook, 1st Edition. Springer, 2011.

S. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, A. Payne. Okapi at TREC-4. 4th Text Retrieval Conference (TREC 1995). Gaithersburg, Maryland, USA, November 1995.

M. Rosvall, C. T. Bergstrom. Maps of Random Walks on Complex Networks Reveal Community Structure. National Academy of Sciences, 105(4), January 2008, pp. 1118-1123.

R. Santos, C. Macdonald, I. Ounis. Exploiting Query Reformulations for Web Search Result Diversification. 19th International Conference on World Wide Web (WWW 2010). Raleigh, NC, USA, April 2010, pp. 881-890.

B. Sarwar, G. Karypis, J. Konstan, J. Riedl. Application of dimensionality reduction in recommender system-a case study. ACM WebKDD Web Mining for E-Commerce Workshop. Boston, MA, August 2000.

G. Shani, A. Gunawardana. Evaluating Recommendation Systems. Recommender Systems Handbook. Springer, 2011, pp. 257-297.

S. Vargas, P. Castells. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. 5th ACM Conference on Recommender Systems (RecSys 2011). Chicago, Illinois, October 2011, pp. 109-116.

S. Wasserman, K. Faust. Social Network Analysis: Methods and Applications. Cambridge University Press. November 1994.

D. J. Watts, S. H. Strogatz. Collective Dynamics of ‘small-world’ Networks. Nature, 393(6684). June 1998, pp. 440-442.

S. White, P. Smyth. Algorithms for estimating relative importance in networks. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003). Washington, DC, USA, August 2003, pp. 266-275.

C. X. Zhai, W. W. Cohen, J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003). Toronto, Canada, July 2003, pp. 10-17.

X. Zhang, J. Cheng, T. Yuan, B. Niu, H. Lu. TopRec: Domain-Specific Recommendation through Community Topic Mining in Social Network. 22nd International Conference on World Wide Web (WWW 2013). Rio de Janeiro, Brazil, May 2013, pp. 1501-1510.

Anexo I. Resultados completos offline

A continuación se muestran los resultados completos obtenidos en las pruebas offline (Tablas 9-14).

	Accuracy						
Recommender	num_q	P@10	P@20	R@10	R@20	nDCG@10	nDCG@20
HKV (k=100, λ=150, α=40)	7086	**0.0649**	**0.0552**	**0.0921**	**0.1555**	**0.0914**	**0.1098**
User kNN (|N(u)|=50)	7086	0.0619	0.0526	0.0879	0.1458	0.0882	0.1047
BM25 (N(u)=UND, N(v)=UND, b=0.2, k=100)	7086	0.0608	0.0502	0.0919	0.1451	0.0892	0.1041
ExtremeBM25 (N(u)=UND, N(v)=UND, b=0.2)	7086	0.0606	0.0502	0.0916	0.1453	0.0890	0.1042
QLJM (N(u)=OUT, N(v)=IN, λ=0.5)	7086	0.0594	0.0504	0.0861	0.1415	0.0859	0.1019
Item kNN (|N(u)|=700)	7086	0.0580	0.0478	0.0816	0.1313	0.0819	0.0951
Adamic (N(u)=OUT, N(v)=IN, N(w)=IN)	7086	0.0574	0.0494	0.0825	0.1374	0.0814	0.0976
FOAF (N(u)=OUT, N(v)=IN)	7086	0.0556	0.0483	0.0785	0.1329	0.0782	0.0943
Tf-idf (N(u)=OUT, N(v)=IN)	7086	0.0495	0.0425	0.0730	0.1210	0.0726	0.0866
Jaccard (N(u)=OUT, N(v)=IN)	7086	0.0474	0.0403	0.0695	0.1137	0.0691	0.0819
PurePersonalizedPageRank (r=0.8)	7086	0.0450	0.0414	0.0679	0.1189	0.0617	0.0786
PersonalizedPageRank (r=0.8)	7086	0.0447	0.0413	0.0671	0.1188	0.0613	0.0785
Closure (N(u)=OUT, N(v)=IN)	7086	0.0417	0.0282	0.0592	0.0748	0.0683	0.0689
PageRank (r=0.6)	7086	0.0160	0.0126	0.0250	0.0359	0.0239	0.0265
Popularity	7086	0.0149	0.0141	0.0156	0.0293	0.0161	0.0203
SALSA (authorities recommendation)	7086	0.0149	0.0141	0.0156	0.0293	0.0161	0.0203
Rocchio	7086	0.0103	0.0104	0.0109	0.0223	0.0114	0.0154
SALSA (hubs recommendation)	7086	0.0028	0.0029	0.0033	0.0057	0.0029	0.0039
Random	7086	0.0008	0.0009	0.0010	0.0018	0.0010	0.0013

Tabla 9. Resultados completos de acierto obtenidos para todos los algoritmos de recomendación estudiados.

	Graph								
	Leading Vector			Infomap			Louvain		
Recommender	Mod	Bridges	Edges	Mod	Bridges	Edges	Mod	Bridges	Edges
HKV (k=100, λ=150, α=40)	0.6237	15992	15992	0.6677	17427	17427	0.7122	13878	13878
User kNN (|N(u)|=50)	0.6257	15152	15152	0.6790	11660	11660	0.7219	9088	9088
BM25 (N(u)=UND, N(v)=UND, b=0.2, k=100)	0.6268	14733	14733	0.6854	8559	8559	0.7258	7239	7239
ExtremeBM25 (N(u)=UND, N(v)=UND, b=0.2)	0.6269	14719	14719	0.6854	8557	8557	0.7258	7237	7237
QLJM (N(u)=OUT, N(v)=IN, λ=0.5)	0.6256	15245	15245	0.6799	11322	11322	0.7220	9052	9052
Item kNN (|N(u)|=700)	0.6265	14819	14819	0.6816	10250	10250	0.7230	8465	8465
Adamic (N(u)=OUT, N(v)=IN, N(w)=IN)	0.6210	17171	17171	0.6709	15683	15683	0.7145	12650	12650
FOAF (N(u)=OUT, N(v)=IN)	0.6216	16817	16817	0.6696	16253	16253	0.7134	13181	13181
Tf-idf (N(u)=OUT, N(v)=IN)	0.6229	16577	16577	0.6779	12718	12718	0.7195	10448	10448
Jaccard (N(u)=OUT, N(v)=IN)	0.6255	15366	15366	0.6811	10936	10936	0.7223	9017	9017
PurePersonalizedPageRank (r=0.8)	0.6078	23287	23287	0.6496	27122	27122	0.6954	22424	22424
PersonalizedPageRank (r=0.8)	0.6077	23304	23304	0.6494	27194	27194	0.6952	22500	22500
Closure (N(u)=OUT, N(v)=IN)	0.5881	33376	33376	0.6362	34906	34906	0.6798	30430	30430
PageRank (r=0.6)	0.5530	47395	47395	0.5909	56162	56162	0.6288	55309	55309
Popularity	0.5420	53166	53166	0.5889	55935	55935	0.6246	56484	56484
SALSA (authorities recommendation)	0.5420	53166	53166	0.5889	55935	55935	0.6246	56484	56484
Rocchio	0.5486	50569	50569	0.5883	59221	59221	0.6283	55592	55592
SALSA (hubs recommendation)	**0.5388**	**55873**	**55873**	**0.5856**	59560	59560	**0.6244**	57107	57107
Random	0.5495	50604	50604	0.5859	**61546**	**61546**	0.6265	**57549**	**57549**

Tabla 10. Resultados completos de diversidad estructural obtenidos para todos los algoritmos de recomendación estudiados (1 de 2).

Recommender	Graph				
	Deg Mod	CCG	CCLA	Arraigo	Arraigo0
HKV (k=100, λ=150, α=40)	-0.2263	0.1330	0.1644	0.0668	1735
User kNN (|N(u)|=50)	-0.2082	0.1376	0.1715	0.0765	1303
BM25 (N(u)=UND, N(v)=UND, b=0.2, k=100)	-0.2136	0.1433	0.1905	0.0899	2705
ExtremeBM25 (N(u)=UND, N(v)=UND, b=0.2)	-0.2143	0.1433	0.1905	0.0900	2682
QLJM (N(u)=OUT, N(v)=IN, λ=0.5)	-0.2039	0.1392	0.1715	0.0995	173
Item kNN (|N(u)|=700)	-0.1952	0.1391	0.1745	0.0814	1447
Adamic (N(u)=OUT, N(v)=IN, N(w)=IN)	-0.1898	0.1317	0.1670	0.0758	160
FOAF (N(u)=OUT, N(v)=IN)	-0.1860	0.1308	0.1644	0.0740	160
Tf-idf (N(u)=OUT, N(v)=IN)	-0.2301	0.1481	0.1732	0.1205	215
Jaccard (N(u)=OUT, N(v)=IN)	-0.2207	0.1463	0.1688	0.1280	217
PurePersonalizedPageRank (r=0.8)	-0.2381	0.1259	0.1552	0.0584	185
PersonalizedPageRank (r=0.8)	-0.2381	0.1258	0.1548	0.0582	185
Closure (N(u)=OUT, N(v)=IN)	-0.2074	0.1430	0.1656	0.0441	29003
PageRank (r=0.6)	-0.2432	0.1041	0.1335	0.0061	17031
Popularity	-0.1637	0.0992	0.1352	0.0069	23808
SALSA (authorities recommendation)	-0.1637	0.0992	0.1352	0.0069	23808
Rocchio	-0.1107	0.0981	0.1450	0.0070	35677
SALSA (hubs recommendation)	**0.3142**	**0.0789**	0.1563	0.0036	51779
Random	-0.2746	0.1078	**0.0927**	**0.0024**	**61908**

Tabla 11. Resultados completos de diversidad estructural obtenidos para todos los algoritmos de recomendación estudiados (2 de 2).

Recommender	Diversity			
	C ERR-IA			HT ERR-IA
	Leading Vector	Infomap	Louvain	
HKV (k=100, λ=150, α=40)	**0.0669**	**0.0643**	0.067079	0.8320
User kNN (|N(u)|=50)	0.0649	0.0624	0.0674	0.8462
BM25 (N(u)=UND, N(v)=UND, b=0.2, k=100)	0.0658	0.0637	**0.0686**	**0.8845**
ExtremeBM25 (N(u)=UND, N(v)=UND, b=0.2)	0.0657	0.0636	0.0686	0.8837
QLJM (N(u)=OUT, N(v)=IN, λ=0.5)	0.0635	0.0622	0.0653	0.8420
Item kNN (|N(u)|=700)	0.0608	0.0603	0.0644	0.8140
Adamic (N(u)=OUT, N(v)=IN, N(w)=IN)	0.0588	0.0568	0.0596	0.7616
FOAF (N(u)=OUT, N(v)=IN)	0.0573	0.0557	0.0579	0.7375
Tf-idf (N(u)=OUT, N(v)=IN)	0.0528	0.0531	0.0573	0.7553
Jaccard (N(u)=OUT, N(v)=IN)	0.0500	0.0502	0.0544	0.7122
PurePersonalizedPageRank (r=0.8)	0.0419	0.0394	0.0431	0.5208
PersonalizedPageRank (r=0.8)	0.0417	0.0392	0.0415	0.5179
Closure (N(u)=OUT, N(v)=IN)	0.0534	0.0529	0.0583	0.7656
PageRank (r=0.6)	0.0154	0.0111	0.0092	0.2296
Popularity	0.0076	0.0082	0.0069	0.1409
SALSA (authorities recommendation)	0.0076	0.0082	0.0071	0.1409
Rocchio	0.0059	0.0062	0.0065	0.0871
SALSA (hubs recommendation)	0.0013	0.0013	0.0017	0.0290
Random	0.0006	0.0005	0.0005	0.0091

Tabla 12. Resultados completos de diversidad obtenidos para todos los algoritmos de recomendación estudiados (1 de 2).

	Diversity					
Recommender	C SR@10	C SR@20	C SRN@10	C SRN@20	Gini	C Gini
HKV (k=100, λ=150, α=40)	**0.1282**	**0.1784**	0.4717	0.5343	0.1207	0.1182
User kNN (|N(u)|=50)	0.1178	0.1632	0.4159	0.4705	0.0999	0.1300
BM25 (N(u)=UND, N(v)=UND, b=0.2, k=100)	0.1194	0.1608	0.4094	0.4576	0.2233	0.1573
ExtremeBM25 (N(u)=UND, N(v)=UND, b=0.2)	0.1192	0.1610	0.4094	0.4573	0.2242	0.1575
QLJM (N(u)=OUT, N(v)=IN, λ=0.5)	0.1165	0.1604	0.4238	0.4803	0.1781	0.1429
Item kNN (|N(u)|=700)	0.1097	0.1507	0.3736	0.4004	0.0967	0.1324
Adamic (N(u)=OUT, N(v)=IN, N(w)=IN)	0.1148	0.1617	0.4520	0.5084	0.0902	0.1163
FOAF (N(u)=OUT, N(v)=IN)	0.1127	0.1587	0.4455	0.5015	0.0773	0.1086
Tf-idf (N(u)=OUT, N(v)=IN)	0.0959	0.1346	0.4369	0.4917	0.4395	0.1787
Jaccard (N(u)=OUT, N(v)=IN)	0.0916	0.1265	0.4174	0.4670	0.3719	0.1686
PurePersonalizedPageRank (r=0.8)	0.1034	0.1534	0.5609	0.6381	0.0760	0.1224
PersonalizedPageRank (r=0.8)	0.1024	0.1530	**0.5611**	0.6377	0.0735	0.1209
Closure (N(u)=OUT, N(v)=IN)	0.0846	0.0988	0.5462	**0.6826**	0.1551	0.1449
PageRank (r=0.6)	0.0339	0.0481	0.3803	0.4207	0.0011	0.0127
Popularity	0.0274	0.0448	0.3718	0.4449	0.0012	0.0082
SALSA (authorities recommendation)	0.0274	0.0448	0.3718	0.4449	0.0012	0.0082
Rocchio	0.0188	0.0365	0.3935	0.5503	0.0031	0.0514
SALSA (hubs recommendation)	0.0059	0.0103	0.4870	0.5503	0.0009	0.0416
Random	0.0017	0.0038	0.5350	0.6713	**0.7906**	**0.2008**

Tabla 13. Resultados completos de diversidad obtenidos para todos los algoritmos de recomendación estudiados (2 de 2).

	Novelty			
Recommender	EPC@10	EPC@20	C EPD@10	C EPD@20
HKV (k=100, λ=150, α=40)	0.9624	0.9657	0.3302	0.3331
User kNN (|N(u)|=50)	0.9579	0.9619	0.3063	0.3155
BM25 (N(u)=UND, N(v)=UND, b=0.2, k=100)	0.9713	0.9736	0.3116	0.3173
ExtremeBM25 (N(u)=UND, N(v)=UND, b=0.2)	0.9713	0.9737	0.3118	0.3172
QLJM (N(u)=OUT, N(v)=IN, λ=0.5)	0.9727	0.9732	0.3081	0.3173
Item kNN (|N(u)|=700)	0.9595	0.9631	0.2859	0.2906
Adamic (N(u)=OUT, N(v)=IN, N(w)=IN)	0.9527	0.9574	0.3243	0.3344
FOAF (N(u)=OUT, N(v)=IN)	0.9518	0.9565	0.3205	0.3309
Tf-idf (N(u)=OUT, N(v)=IN)	0.9924	0.9915	0.3387	0.3432
Jaccard (N(u)=OUT, N(v)=IN)	0.9924	0.9919	0.3218	0.3283
PurePersonalizedPageRank (r=0.8)	0.9583	0.9601	0.4349	0.4241
PersonalizedPageRank (r=0.8)	0.9576	0.9598	0.4354	0.4245
Closure (N(u)=OUT, N(v)=IN)	0.9894	0.9903	0.5711	0.6288
PageRank (r=0.6)	0.8940	0.9063	0.7561	0.7611
Popularity	0.8699	0.8897	0.7551	0.7817
SALSA (authorities recommendation)	0.8699	0.8897	0.7551	0.7817
Rocchio	0.9289	0.9341	0.8208	0.7938
SALSA (hubs recommendation)	0.9606	0.9726	0.8164	0.8072
Random	**0.9944**	**0.9944**	**0.8549**	**0.8548**

Tabla 14. Resultados completos de novedad obtenidos para todos los algoritmos de recomendación estudiados.

Anexo II. Significatividad estadística

A continuación se muestran los resultados de la significatividad estadística realizada para todos los algoritmos sobre los valores de la precisión @10 de cada uno de los usuarios recomendados (Tabla 15).

	HKV	IkNN	Adam.	BM25	Clos.	EBM25	FOAF	Jacc.	PR	PPR	Pop.	PPPR	QLJM	Roc.	SLSA(A)	SLSA(H)	TFIDF	Rnd
IkNN	0.00																	
Adam.	0.00	**0.22**																
BM25	0.00	0.00	0.00															
Clos.	0.00	0.00	0.00	0.00														
EBM25	0.00	0.00	0.00	**0.06**	0.00													
FOAF	0.00	0.00	0.00	0.00	0.00	0.00												
Jacc.	0.00	0.00	0.00	0.00	0.00	0.00	0.00											
PR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
PPR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00									
Pop.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00								
PPPR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00							
QLJM	0.00	**0.05**	0.00	0.05	0.00	**0.06**	0.00	0.00	0.00	0.00	0.00	0.00						
Roc.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00					
SLSA(A)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	-	0.00	0.00	0.00				
SLSA(H)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			
TFIDF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
Rnd	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
UkNN	0.00	0.00	0.00	**0.09**	0.00	**0.07**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Tabla 15. Resultados de significatividad estadística para Precisión @10 obtenidos entre todos los algoritmos de recomendación estudiados.